

The sTeX3 Package Collection *

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

2022-04-25

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 2022-04-25)

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	92
24.3	Messages and logging	93
24.4	HTML Annotations	94
24.5	Babel Languages	95
24.6	Persistence	96
24.7	Auxiliary Methods	97
25	<code>sTeX</code>-MathHub Implementation	100
25.1	Generic Path Handling	100
25.2	PWD and <code>kpsewhich</code>	102
25.3	File Hooks and Tracking	103
25.4	MathHub Repositories	104
25.5	Using Content in Archives	109
26	<code>sTeX</code>-References Implementation	113
26.1	Document URIs and URLs	113
26.2	Setting Reference Targets	115
26.3	Using References	117
27	<code>sTeX</code>-Modules Implementation	120
27.1	The <code>smodule</code> environment	124
27.2	Invoking modules	130

28	STEX-Module Inheritance Implementation	132
28.1	SMS Mode	132
28.2	Inheritance	136
29	STEX-Symbols Implementation	141
29.1	Symbol Declarations	141
29.2	Notations	148
29.3	Variables	157
30	STEX-Terms Implementation	163
30.1	Symbol Invocations	163
30.2	Terms	170
30.3	Notation Components	174
30.4	Variables	176
30.5	Sequences	178
31	STEX-Structural Features Implementation	179
31.1	Imports with modification	180
31.2	The feature environment	188
31.3	Structure	188
32	STEX-Statements Implementation	198
32.1	Definitions	198
32.2	Assertions	203
32.3	Examples	207
32.4	Logical Paragraphs	209
33	The Implementation	215
33.1	Proofs	215
33.2	Justifications	226
34	STEX-Others Implementation	227
35	STEX-Metattheory Implementation	228
36	Tikzinput Implementation	231
37	document-structure.sty Implementation	234
37.1	Package Options	234
37.2	Document Structure	235
37.3	Front and Backmatter	239
37.4	Global Variables	241
38	NotesSlides – Implementation	242
38.1	Class and Package Options	242
38.2	Notes and Slides	244
38.3	Header and Footer Lines	248
38.4	Frame Images	250
38.5	Colors and Highlighting	251
38.6	Sectioning	252
38.7	Excursions	254

39 The Implementation	256
39.1 Package Options	256
39.2 Problems and Solutions	257
39.3 Multiple Choice Blocks	264
39.4 Including Problems	265
39.5 Reporting Metadata	267
40 Implementation: The hwexam Package	269
40.1 Package Options	269
40.2 Assignments	270
40.3 Including Assignments	273
40.4 Typesetting Exams	274
40.5 Leftovers	276
41 References	277

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 sTeX Document

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

TODO: use some sTeX-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>
```

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?infinitesum"/>
  <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 ($\langle\textit{language}\rangle*$) Languages to load with the `babel` package.

mathhub ($\langle\textit{directory}\rangle$) MathHub folder to search for repositories – this is not necessary if the `MATHHUB` system variable is set.

sms ($\langle\textit{boolean}\rangle$) use *persisted* mode (not yet implemented).

image ($\langle\textit{boolean}\rangle$) passed on to `tikzinput`.

debug ($\langle\textit{log-prefix}\rangle*$) 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.

$\hookrightarrow M \rightarrow$

$\hookrightarrow M \rightarrow$

$\hookrightarrow T \rightarrow$

- 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 `\smodulename`, `\smoduletype` and `\smoduleid`.

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

\hookrightarrow `\symdecl` introduces a new OMDoc/MMT constant in the current module (\Rightarrow 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

-  $M \rightarrow$ Applications of semantic macros, such as `\binarysymbol{a}{b}` are translated to
-  $M \rightarrow$ MMT/OMDOC as OMA-terms with head `<OMS name="...?binarysymbol"/>`.
-  $T \rightarrow$ 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}^{\varn}}
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}{rminus}
7   \end{copymodule}
8   \notation*{rplus}[plus,op=+,prec=60]{\#1 \comp+ \#2}
9   \notation*{rzero}[zero]{\comp0}
10  \notation*{rminus}[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 $\text{\texttt{STEX}}$ Metatheory)

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

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

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

We make this theory part of the $\text{\texttt{STEX}}$ collection rather than encoding it in $\text{\texttt{STEX}}$ 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
 \rightsquigarrow T \rightsquigarrow `<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  $\text{\addition{\svar{n}}{\svar{m}}}$ , then
2  $\text{\addition*{}$ 
3    $\text{\arg*{\addition{\svar{n}}{\svar{m}}}}$ 
4    $\text{\comp{+}}$ 
5    $\text{\arg{\svar{k}}}$ 
6  $\text{}}$  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 `tikzinput` package is determined by whether the `image` option is given. If it is not, then the `tikz` package is loaded, all other options are passed on to it and `\tikzinput{<file>}` inputs the TIKZ file `<file>.tex`; if not, only the `graphicx` package is loaded and `\tikzinput{<file>}` loads an image file `<file>.<ext>` generated from `<file>.tex`.

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

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

The `standalone` class is a minimal \LaTeX class that when loaded in a document that uses the `standalone` package: the preamble and the `document` environment are disregarded during loading, so they do not pose any problems. In effect, an `\input` of the file above only sees the `tikzpicture` environment, but the file itself is standalone in the sense that we can run \LaTeX over it separately, e.g. for generating an image file from it.

\tikzinput
\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>}{<ctext>}` tests the content of the global variable `<vname>`, only if (after expansion) it is equal to `<val>`, the conditional text `<ctext>` 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=Noooooooooooo,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)

2022-04-25

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 X _{ML}
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E X _{ML} .
<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_EX_{ML} or R_US_TE_X) with attributes:

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

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

<hr/> <hr/> <code>\c_stex_module_<URI>_prop</code>	<p>A property list with the following fields:</p> <p>name The <i>name</i> of the module,</p> <p>ns the <i>namespace</i> in field ns,</p> <p>file the <i>file</i> containing the module, as a sequence of path fragments</p> <p>lang the module's <i>language</i>,</p> <p>sig the language of the signature module, if the current file is a translation from some other language,</p> <p>deprecate if this module is deprecated, the module that replaces it,</p> <p>meta the metatheory of the module.</p>
--	---

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

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

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

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

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` 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/>	$\text{\notation}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <p>Introduces a new notation for $\langle symbol \rangle$, see <code>\stex_notation_do:nn</code></p>
<hr/> <hr/> <code>\stex_notation_do:nn</code>	$\text{\stex_notation_do:nn}\{\langle URI \rangle\}\{\langle notations^+ \rangle\}$ <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/>	$\text{\symdef}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <p>Combines <code>\symdecl</code> and <code>\notation</code> by introducing a new symbol and assigning a new notation for it.</p>

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

<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 \SIX 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 \SIX 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

sT_EX-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

sTeX -Basics Implementation

24.1 The sTeXDocument 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 </cls>
```

24.2 Preliminaries

```
23 <*package>
24
25 %%%%%%%%% basics.dtx %%%%%%%%%
26
```

```

27 \RequirePackage{expl3,l3keys2e,ltxcmds}
28 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
29
30 \bool_if_exist:NF \c_stex_document_class_bool {
31   \bool_set_false:N \c_stex_document_class_bool
32   \RequirePackage{standalone}
33 }
34
35 \message{^^J
36   *****^^J
37   *~This~is~sTeX~version~3.1.0~*^^J
38   *****^^J
39 ^^J}
40
41 %\RequirePackage{morewrites}
42 %\RequirePackage{amsmath}
43
44 Package options:
45 \keys_define:nn { stex } {
46   debug      .clist_set:N = \c_stex_debug_clist ,
47   lang       .clist_set:N = \c_stex_languages_clist ,
48   mathhub    .tl_set_x:N = \mathhub ,
49   usesms     .bool_set:N = \c_stex_persist_mode_bool ,
50   writesms   .bool_set:N = \c_stex_persist_write_mode_bool ,
51   image      .bool_set:N = \c_tikzinput_image_bool ,
52   unknown    .code:n      = {}
53 }
54 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX

```

54 \RequirePackage{xspace}
55 \protected\def\stex{
56   \@ifundefined{texorpdfstring}{\let\texorpdfstring\@firstoftwo}{\let\texorpdfstring{\raisebox{- .5ex}{S\kern-.5ex\TeX}}{sTeX}\xspace}
57 }
58 \let\sTeX\stex

```

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

24.3 Messages and logging

```

60 <@@=stex_log>

```

Warnings and error messages

```

61 \msg_new:nnn{stex}{error/unknownlanguage}{
62   Unknown~language:~#1
63 }
64 \msg_new:nnn{stex}{warning/nomathhub}{
65   MATHHUB~system~variable~not~found~and~no~
66   \detokenize{\mathhub}~value~set!
67 }
68 \msg_new:nnn{stex}{error/deactivated-macro}{
69   The~\detokenize{#1}~command~is~only~allowed~in~#2!
70 }

```

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

```

71 \cs_new_protected:Nn \stex_debug:nn {
72   \clist_if_in:NnTF \c_stex_debug_clist { all } {
73     \msg_set:nnn{stex}{debug / #1}{
74       \\Debug~#1:~#2\\
75     }
76     \msg_none:nn{stex}{debug / #1}
77   }{
78     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
79       \msg_set:nnn{stex}{debug / #1}{
80         \\Debug~#1:~#2\\
81       }
82       \msg_none:nn{stex}{debug / #1}
83     }
84   }
85 }

```

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

Redirecting messages:

```

86 \clist_if_in:NnTF \c_stex_debug_clist {all} {
87   \msg_redirect_module:nnn{ stex }{ none }{ term }
88 }{
89   \clist_map_inline:Nn \c_stex_debug_clist {
90     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
91   }
92 }
93
94 \stex_debug:nn{log}{debug~mode~on}

```

24.4 HTML Annotations

95 <@@=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

96 \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

```

97 \cs_new_protected:Nn \_stex_html_checkempty:n {
98   \tl_set:Nn \l_stex_html_arg_tl { #1 }
99   \tl_if_empty:NT \l_stex_html_arg_tl {
100     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
101   }
102 }

```

(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

```

103 \bool_new:N \_stex_html_do_output_bool
104 \bool_set_true:N \_stex_html_do_output_bool
105

```

```

106 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
107   \bool_if:nTF \_stex_html_do_output_bool
108     \prg_return_true: \prg_return_false:
109 }

```

(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

```

110 \cs_new_protected:Nn \stex_suppress_html:n {
111   \exp_args:Nne \use:nn {
112     \bool_set_false:N \_stex_html_do_output_bool
113     #1
114   }{
115     \stex_if_do_html:T {
116       \bool_set_true:N \_stex_html_do_output_bool
117     }
118   }
119 }

```

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

`\stex_annotate:enw`

`\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.

```

120 \tl_if_exist:NF\stex@backend{
121   \ifcsname if@rustex\endcsname
122     \def\stex@backend{rustex}
123   \else
124     \ifcsname if@latexml\endcsname
125       \def\stex@backend{latexml}
126     \else
127       \def\stex@backend{pdflatex}
128     \fi
129   \fi
130 }
131 \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

```

132 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

133 \prop_const_from_keyval:Nn \c_stex_languages_prop {
134   en = english ,
135   de = ngerman ,
136   ar = arabic ,
137   bg = bulgarian ,
138   ru = russian ,
139   fi = finnish ,
140   ro = romanian ,

```

```

141   tr = turkish ,
142   fr = french
143 }
144
145 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
146   english   = en ,
147   ngerman   = de ,
148   arabic    = ar ,
149   bulgarian = bg ,
150   russian   = ru ,
151   finnish   = fi ,
152   romanian  = ro ,
153   turkish   = tr ,
154   french    = fr
155 }
156 % todo: chinese simplified (zhs)
157 %       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:

```

158 \clist_if_empty:NF \c_stex_languages_clist {
159   \clist_clear:N \l_tmpa_clist
160   \clist_map_inline:Nn \c_stex_languages_clist {
161     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
162       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
163     } {
164       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
165     }
166   }
167   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
168   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
169 }
170
171 \AtBeginDocument{
172   \stex_html_backend:T {
173     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
174     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
175     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
176     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
177     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
178       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
179       \stex_debug:nn{basics} {Language~\l_tmpa_str~
180         inferred~from~file~name}
181       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
182     }
183   }
184 }

```

24.6 Persistence

```

185 <@@=stex_persist>
186 \bool_if:NTF \c_stex_persist_mode_bool {

```

```

187 \def \stex_persist:n #1 {}
188 \def \stex_persist:x #1 {}
189 }{
190 \bool_if:NTF \c_stex_persist_write_mode_bool {
191 \iow_new:N \c__stex_persist_iow
192 \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
193 \AtEndDocument{
194 \iow_close:N \c__stex_persist_iow
195 }
196 \cs_new_protected:Nn \stex_persist:n {
197 \tl_set:Nn \l_tmpa_tl { #1 }
198 \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
199 \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
200 }
201 \cs_generate_variant:Nn \stex_persist:n {x}
202 }{
203 \def \stex_persist:n #1 {}
204 \def \stex_persist:x #1 {}
205 }
206 }

```

24.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

207 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
208 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
209 \def#1{
210 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
211 }
212 }

```

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

\stex_reactivate_macro:N

```

213 \cs_new_protected:Nn \stex_reactivate_macro:N {
214 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
215 }

```

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

\ignorespacesandpars

```

216 \protected\def\ignorespacesandpars{
217 \begingroup\catcode13=10\relax
218 \@ifnextchar\par{
219 \endgroup\expandafter\ignorespacesandpars\@gobble
220 }{
221 \endgroup
222 }
223 }
224
225 \cs_new_protected:Nn \stex_copy_control_sequence:NNN {
226 \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
227 \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
228 \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}

```

```

229
230 \tl_clear:N \_tmp_args_tl
231 \int_step_inline:nn \l_tmpa_int {
232   \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
233 }
234
235 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
236 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
237   \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
238   \exp_after:wN\exp_after:wN\exp_after:wN {
239     \exp_after:wN #2 \_tmp_args_tl
240   }
241 }}
242 }
243 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
244 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
245 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
246
247 \cs_new_protected:Nn \stex_copy_control_sequence_ii:NNN {
248   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}
249   \exp_args:NNo \tl_remove_all:Nn \_tmp_args_tl \c_hash_str
250   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}
251
252   \tl_clear:N \_tmp_args_tl
253   \int_step_inline:nn \l_tmpa_int {
254     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{#####}\exp_not:n{##1}}}
255   }
256
257   \edef \_tmp_args_tl {
258     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
259     \exp_after:wN\exp_after:wN\exp_after:wN {
260       \exp_after:wN #2 \_tmp_args_tl
261     }
262   }
263
264   \exp_after:wN \def \exp_after:wN \_tmp_args_tl
265   \exp_after:wN ##\exp_after:wN 1 \exp_after:wN ##\exp_after:wN 2
266   \exp_after:wN { \_tmp_args_tl }
267
268   \edef \_tmp_args_tl {
269     \exp_after:wN \exp_not:n \exp_after:wN {
270       \_tmp_args_tl {####1}{####2}
271     }
272   }
273
274   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
275   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
276     \exp_after:wN\exp_not:n\exp_after:wN{\_tmp_args_tl}
277   }}
278 }
279
280 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {cNN}
281 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {NcN}
282 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}

```


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

`\MMTrule`

```

283 \NewDocumentCommand \MMTrule {m m}{
284   \seq_set_split:Nnn \l_tmpa_seq , {#2}
285   \int_zero:N \l_tmpa_int
286   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
287     \seq_if_empty:NF \l_tmpa_seq {
288       $\seq_map_inline:Nn \l_tmpa_seq {
289         \int_incr:N \l_tmpa_int
290         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
291       }$
292     }
293   }
294 }
295
296 \NewDocumentCommand \MMTinclude {m}{
297   \stex_annotate_invisible:nnn{import}{#1}{ }
298 }
299
300 \tl_new:N \g_stex_document_title
301 \cs_new_protected:Npn \STEXtitle #1 {
302   \tl_if_empty:NT \g_stex_document_title {
303     \tl_gset:Nn \g_stex_document_title { #1 }
304   }
305 }
306 \cs_new_protected:Nn \stex_document_title:n {
307   \tl_if_empty:NT \g_stex_document_title {
308     \tl_gset:Nn \g_stex_document_title { #1 }
309     \stex_annotate_invisible:n{\noindent
310       \stex_annotate:nnn{doctitle}{ }{ #1 }
311     \par}
312   }
313 }
314 \AtBeginDocument {
315   \let \STEXtitle \stex_document_title:n
316   \tl_if_empty:NF \g_stex_document_title {
317     \stex_annotate_invisible:n{\noindent
318       \stex_annotate:nnn{doctitle}{ }{ \g_stex_document_title }
319     \par}
320   }
321 }
322
323 </package>

```

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

Chapter 25

STEX -MathHub Implementation

```
324 <*package>
325
326 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
327
328 <@@=stex_path>
329
330 Warnings and error messages
331 \msg_new:nnn{stex}{error/norepository}{
332   No~archive~#1~found~in~#2
333 }
334 \msg_new:nnn{stex}{error/notinarchive}{
335   Not~currently~in~an~archive,~but~\detokenize{#1}~
336   needs~one!
337 }
338 \msg_new:nnn{stex}{error/nofile}{
339   \detokenize{#1}~could~not~find~file~#2
340 }
341 \msg_new:nnn{stex}{error/twofiles}{
342   \detokenize{#1}~found~two~candidates~for~#2
343 }
```

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`

```
342 \cs_new_protected:Nn \stex_path_from_string:Nn {
343   \str_set:Nx \l_tmpa_str { #2 }
344   \str_if_empty:NTF \l_tmpa_str {
345     \seq_clear:N #1
346   }{
347     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
348     \sys_if_platform_windows:T{
349       \seq_clear:N \l_tmpa_tl
```

```

350     \seq_map_inline:Nn #1 {
351       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
352       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
353     }
354     \seq_set_eq:NN #1 \l_tmpa_tl
355   }
356   \stex_path_canonicalize:N #1
357 }
358 }
359

```

(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`

```

360 \cs_new_protected:Nn \stex_path_to_string:NN {
361   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
362 }
363
364 \cs_new:Nn \stex_path_to_string:N {
365   \seq_use:Nn #1 /
366 }

```

(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`

```

367 \str_const:Nn \c__stex_path_dot_str {.}
368 \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.

```

369 \cs_new_protected:Nn \stex_path_canonicalize:N {
370   \seq_if_empty:NF #1 {
371     \seq_clear:N \l_tmpa_seq
372     \seq_get_left:NN #1 \l_tmpa_tl
373     \str_if_empty:NT \l_tmpa_tl {
374       \seq_put_right:Nn \l_tmpa_seq {}
375     }
376     \seq_map_inline:Nn #1 {
377       \str_set:Nn \l_tmpa_tl { ##1 }
378       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
379         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
380           \seq_if_empty:NNTF \l_tmpa_seq {
381             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
382               \c__stex_path_up_str
383             }
384           }{
385             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
386             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
387               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
388                 \c__stex_path_up_str
389               }
390             }{

```

```

391         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
392     }
393 }
394 }{
395     \str_if_empty:NF \l_tmpa_tl {
396         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
397     }
398 }
399 }
400 }
401 \seq_gset_eq:NN #1 \l_tmpa_seq
402 }
403 }

```

(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}`

```

404 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
405     \seq_if_empty:NTF #1 {
406         \prg_return_false:
407     }{
408         \seq_get_left:NN #1 \l_tmpa_tl
409         \sys_if_platform_windows:TF{
410             \str_if_in:NnTF \l_tmpa_tl {:}{
411                 \prg_return_true:
412             }{
413                 \prg_return_false:
414             }
415         }{
416             \str_if_empty:NTF \l_tmpa_tl {
417                 \prg_return_true:
418             }{
419                 \prg_return_false:
420             }
421         }
422     }
423 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

424 \str_new:N\l_stex_kpsewhich_return_str
425 \cs_new_protected:Nn \stex_kpsewhich:n {
426     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
427     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
428     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
429 }

```

(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`

```

430 \sys_if_platform_windows:TF{
431   \begingroup\escapechar=-1\catcode'\=12
432   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
433   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
434   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
435   }}{
436   \stex_kpsewhich:n{-var-value~PWD}
437 }
438
439 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
440 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
441 \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

```

442 <@@=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

```

443 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

444 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
445 \stex_path_from_string:Nn \c_stex_mainfile_seq
446   \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`

```

447 \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`

```

448 \cs_new_protected:Nn \stex_filestack_push:n {
449   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
450   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
451     \stex_path_from_string:Nn\g_stex_currentfile_seq{
452       \c_stex_pwd_str/#1
453     }
454   }
455   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
456   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
457 }

```

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

`\stex_filestack_pop:`

```

458 \cs_new_protected:Nn \stex_filestack_pop: {
459   \seq_if_empty:NF\g__stex_files_stack{
460     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
461   }
462   \seq_if_empty:NTF\g__stex_files_stack{
463     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
464   }{
465     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
466     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
467   }
468 }
```

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

Hooks for the current file:

```

469 \AddToHook{file/before}{
470   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
471 }
472 \AddToHook{file/after}{
473   \stex_filestack_pop:
474 }
```

25.4 MathHub Repositories

475 `<@@=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.

```

476 \str_if_empty:NTF\mathhub{
477   \sys_if_platform_windows:TF{
478     \begingroup\escapechar=-1\catcode'\=12
479     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
480     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
481     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
482   }{
483     \stex_kpsewhich:n{-var-value-MATHHUB}
484   }
485   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
486 }
487 \str_if_empty:NT \c_stex_mathhub_str {
488   \sys_if_platform_windows:TF{
489     \begingroup\escapechar=-1\catcode'\=12
490     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
491     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
492     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
493   }{
494     \stex_kpsewhich:n{-var-value-HOME}
495   }
496   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
497     \begingroup\escapechar=-1\catcode'\=12
498     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```

499     \sys_if_platform_windows:T{
500       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
501     }
502     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
503     \endgroup
504     \ior_close:N \l_tmpa_ior
505   }
506 }
507 \str_if_empty:NTF\c_stex_mathhub_str{
508   \msg_warning:nn{stex}{warning/nomathhub}
509 }{
510   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
511   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
512 }
513 }{
514   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
515   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
516     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
517       \c_stex_pwd_str/\mathhub
518     }
519   }
520   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
521   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
522 }

```

(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

```

523 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
524   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
525     \str_set:Nx \l_tmpa_str { #1 }
526     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
527     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
528     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
529     \_stex_mathhub_find_manifest:N \l_tmpa_seq
530     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
531       \msg_error:nnxx{stex}{error/norepository}{#1}{
532         \stex_path_to_string:N \c_stex_mathhub_str
533       }
534     } {
535       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
536     }
537   }
538 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```

539 \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`:

```

540 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
541   \seq_set_eq:NN \l_tmpa_seq #1
542   \bool_set_true:N \l_tmpa_bool
543   \bool_while_do:Nn \l_tmpa_bool {
544     \seq_if_empty:NTF \l_tmpa_seq {
545       \bool_set_false:N \l_tmpa_bool
546     }{
547       \file_if_exist:nTF{
548         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
549       }{
550         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
551         \bool_set_false:N \l_tmpa_bool
552       }{
553         \file_if_exist:nTF{
554           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
555         }{
556           \seq_put_right:Nn \l_tmpa_seq{META-INF}
557           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
558           \bool_set_false:N \l_tmpa_bool
559         }{
560           \file_if_exist:nTF{
561             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
562           }{
563             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
564             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
565             \bool_set_false:N \l_tmpa_bool
566           }{
567             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
568           }
569         }
570       }
571     }
572   }
573   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
574 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

575 \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:

```

576 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
577   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
578   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
579   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
580     \str_set:Nn \l_tmpa_str {##1}
581     \exp_args:NNoo \seq_set_split:Nnn
582       \l_tmpb_seq \c_colon_str \l_tmpa_str
583     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```



```

584 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
585 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
586 }
587 \exp_args:No \str_case:nnTF \l_tmpa_tl {
588 {id} {
589 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
590 { id } \l_tmpb_tl
591 }
592 {narration-base} {
593 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
594 { narr } \l_tmpb_tl
595 }
596 {url-base} {
597 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
598 { docurl } \l_tmpb_tl
599 }
600 {source-base} {
601 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
602 { ns } \l_tmpb_tl
603 }
604 {ns} {
605 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
606 { ns } \l_tmpb_tl
607 }
608 {dependencies} {
609 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
610 { deps } \l_tmpb_tl
611 }
612 }{}{}
613 }{}
614 }
615 \ior_close:N \c__stex_mathhub_manifest_ior
616 \stex_persist:x {
617 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
618 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
619 }
620 }
621 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

622 \cs_new_protected:Nn \stex_set_current_repository:n {
623 \stex_require_repository:n { #1 }
624 \prop_set_eq:Nc \l_stex_current_repository_prop {
625 c_stex_mathhub_#1_manifest_prop
626 }
627 }

```

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

`\stex_require_repository:n`

```

628 \cs_new_protected:Nn \stex_require_repository:n {
629 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
630 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

631   \_stex_mathhub_do_manifest:n { #1 }
632   }
633 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

634 %\prop_new:N \l_stex_current_repository_prop
635 \bool_if:NF \c_stex_persist_mode_bool {
636   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
637   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
638     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
639   } {
640     \_stex_mathhub_parse_manifest:n { main }
641     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
642     \l_tmpa_str
643     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
644     \c_stex_mathhub_main_manifest_prop
645     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
646     \stex_debug:nn{mathhub}{Current~repository:~
647     \prop_item:Nn \l_stex_current_repository_prop {id}
648   }
649 }
650 }

```

(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.

```

651 \cs_new_protected:Nn \stex_in_repository:nn {
652   \str_set:Nx \l_tmpa_str { #1 }
653   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
654   \str_if_empty:NTF \l_tmpa_str {
655     \prop_if_exist:NTF \l_stex_current_repository_prop {
656       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
657       \exp_args:Ne \l_tmpa_cs{
658         \prop_item:Nn \l_stex_current_repository_prop { id }
659       }
660     }{
661       \l_tmpa_cs{}
662     }
663   }{
664     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
665     \stex_require_repository:n \l_tmpa_str
666     \str_set:Nx \l_tmpa_str { #1 }
667     \exp_args:Nne \use:nn {
668       \stex_set_current_repository:n \l_tmpa_str
669       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
670     }{
671       \stex_debug:nn{mathhub}{switching~back~to:~
672       \prop_if_exist:NTF \l_stex_current_repository_prop {
673         \prop_item:Nn \l_stex_current_repository_prop { id }::~
674       \meaning\l_stex_current_repository_prop
675     }{

```

```

676         no~repository
677     }
678 }
679 \prop_if_exist:NTF \l_stex_current_repository_prop {
680     \stex_set_current_repository:n {
681         \prop_item:Nn \l_stex_current_repository_prop { id }
682     }
683 }{
684     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
685 }
686 }
687 }
688 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

689 \def \mhpath #1 #2 {
690     \exp_args:Ne \tl_if_empty:nTF{#1}{
691         \c_stex_mathhub_str /
692         \prop_item:Nn \l_stex_current_repository_prop { id }
693         / source / #2
694     }{
695         \c_stex_mathhub_str / #1 / source / #2
696     }
697 }

```

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

`\inputref`

`\mhinput`

```

698 \newif \ifinputref \inputreffalse
699
700 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
701     \stex_in_repository:nn {#1} {
702         \ifinputref
703             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
704         \else
705             \inputreftrue
706             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
707             \inputreffalse
708         \fi
709     }
710 }
711 \NewDocumentCommand \mhinput { 0{} m }{
712     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
713 }
714
715 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
716     \stex_in_repository:nn {#1} {
717         \stex_html_backend:TF {
718             \str_clear:N \l_tmpa_str

```

```

719     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
720       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
721     }
722     \stex_annotate_invisible:nnn{inputref}{
723       \l_tmpa_str / #2
724     }{}
725   }{
726     \begingroup
727     \inputreftrue
728     \tl_if_empty:nTF{ ##1 }{
729       \input{#2}
730     }{
731       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
732     }
733     \endgroup
734   }
735 }
736 }
737 \NewDocumentCommand \inputref { 0{ } m }{
738   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
739 }

```

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

`\addmhbibresource`

```

740 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
741   \stex_in_repository:nn {#1} {
742     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
743   }
744 }
745 \newcommand\addmhbibresource[2][]{
746   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
747 }

```

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

`\libinput`

```

748 \cs_new_protected:Npn \libinput #1 {
749   \prop_if_exist:NF \l_stex_current_repository_prop {
750     \msg_error:nnn{stex}{error/notinarchive}\libinput
751   }
752   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
753     \msg_error:nnn{stex}{error/notinarchive}\libinput
754   }
755   \seq_clear:N \l__stex_mathhub_libinput_files_seq
756   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
757   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
758
759   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
760     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
761     \IfFileExists{ \l_tmpa_str }{
762       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
763     }{}
764     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
765     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

766 }
767
768 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
769 \IfFileExists{ \l_tmpa_str }{
770   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
771 }{}
772
773 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
774   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
775 }{
776   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
777     \input{ ##1 }
778   }
779 }
780 }

```

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

`\libusepackage`

```

781 \NewDocumentCommand \libusepackage {0{} m} {
782   \prop_if_exist:NF \l_stex_current_repository_prop {
783     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
784   }
785   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
786     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
787   }
788   \seq_clear:N \l__stex_mathhub_libinput_files_seq
789   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
790   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
791
792   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
793     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
794     \IfFileExists{ \l_tmpa_str.sty }{
795       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
796     }{}
797     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
798     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
799   }
800
801   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
802   \IfFileExists{ \l_tmpa_str.sty }{
803     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
804   }{}
805
806   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
807     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
808   }{
809     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
810       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
811         \usepackage[#1]{ ##1 }
812       }
813     }{
814       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
815     }
816   }

```

```

816 }
817 }

```

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

```

\mhgraphics
\cmhgraphics

```

```

818
819 \AddToHook{begindocument}{
820 \ltx@ifpackageloaded{graphicx}{
821   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
822   \newcommand\mhgraphics[2][]{%
823     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
824     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
825   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
826 }{}

```

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

```

\lstinputmhlisting
\clstinputmhlisting

```

```

827 \ltx@ifpackageloaded{listings}{
828   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
829   \newcommand\lstinputmhlisting[2][]{%
830     \def\lst@mhrepos{}\setkeys{lst}{#1}%
831     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
832   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
833 }{}
834 }
835
836 \</package>

```

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

Chapter 26

STEX -References Implementation

```
837 <*package>
838
839 %%%%%%%%%% references.dtx %%%%%%%%%%
840
841 <@@=stex_refs>
      Warnings and error messages
842
```

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

```
843 %\iow_new:N \c__stex_refs_refs_iow
844 \AtBeginDocument{
845 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
846 }
847 \AtEndDocument{
848 % \iow_close:N \c__stex_refs_refs_iow
849 }
```

`\STEXreftitle`

```
850 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
851
852 \NewDocumentCommand \STEXreftitle { m } {
853   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
854 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
855 \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:`

```
856 \cs_new_protected:Nn \stex_get_document_uri: {
857   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
858   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
859   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
860   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
861   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
862
863   \str_clear:N \l_tmpa_str
864   \prop_if_exist:NT \l_stex_current_repository_prop {
865     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
866       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
867     }
868   }
869
870   \str_if_empty:NTF \l_tmpa_str {
871     \str_set:Nx \l_stex_current_docns_str {
872       file:/\stex_path_to_string:N \l_tmpa_seq
873     }
874   }{
875     \bool_set_true:N \l_tmpa_bool
876     \bool_while_do:Nn \l_tmpa_bool {
877       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
878       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
879         {source} { \bool_set_false:N \l_tmpa_bool }
880       }{}{
881         \seq_if_empty:NT \l_tmpa_seq {
882           \bool_set_false:N \l_tmpa_bool
883         }
884       }
885     }
886
887     \seq_if_empty:NTF \l_tmpa_seq {
888       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
889     }{
890       \str_set:Nx \l_stex_current_docns_str {
891         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
892       }
893     }
894   }
895 }
```

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

`\l_stex_current_docurl_str`

```
896 \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:`

```
897 \cs_new_protected:Nn \stex_get_document_url: {
898   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
899   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
900   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```



```

901 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
902 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
903
904 \str_clear:N \l_tmpa_str
905 \prop_if_exist:NT \l_stex_current_repository_prop {
906   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
907     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
908       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
909     }
910   }
911 }
912
913 \str_if_empty:NTF \l_tmpa_str {
914   \str_set:Nx \l_stex_current_docurl_str {
915     file:/\stex_path_to_string:N \l_tmpa_seq
916   }
917 }{
918   \bool_set_true:N \l_tmpa_bool
919   \bool_while_do:Nn \l_tmpa_bool {
920     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
921     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
922       {source} { \bool_set_false:N \l_tmpa_bool }
923     }{}{
924       \seq_if_empty:NT \l_tmpa_seq {
925         \bool_set_false:N \l_tmpa_bool
926       }
927     }
928   }
929 }
930 \seq_if_empty:NTF \l_tmpa_seq {
931   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
932 }{
933   \str_set:Nx \l_stex_current_docurl_str {
934     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
935   }
936 }
937 }
938 }

```

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

26.2 Setting Reference Targets

```

939 \str_const:Nn \c__stex_refs_url_str{URL}
940 \str_const:Nn \c__stex_refs_ref_str{REF}
941 \str_new:N \l__stex_refs_curr_label_str
942 % @currentlabel -> number
943 % @currentlabelname -> title
944 % @currentHref -> name.number <- id of some kind
945 % \theH# -> \arabic{section}
946 % \the# -> number
947 % \hyper@makecurrent{#}
948 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

949 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
950   \stex_get_document_uri:
951   \str_clear:N \l__stex_refs_curr_label_str
952   \str_set:Nx \l_tmpa_str { #1 }
953   \str_if_empty:NT \l_tmpa_str {
954     \int_incr:N \l__stex_refs_unnamed_counter_int
955     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
956   }
957   \str_set:Nx \l__stex_refs_curr_label_str {
958     \l_stex_current_docns_str?\l_tmpa_str
959   }
960   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
961     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
962   }
963   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
964     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
965   }
966   \stex_if_smsmode:TF {
967     \stex_get_document_url:
968     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
969     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
970   }{
971     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
972     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
973     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
974     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
975   }
976 }

```

(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.

```

977 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
978   \str_set:Nn \l_tmpa_str {#1?#2}
979   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
980   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
981     \seq_new:c {g__stex_refs_labels_#2_seq}
982   }
983   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
984     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
985   }
986 }

```

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

```

987 \AtEndDocument{
988   \def\stexauxadddocref#1 #2 {}{}
989 }

```

`\stex_ref_new_sym_target:n`

```

990 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
991   \stex_if_smsmode:TF {
992     \str_if_exist:cF{sref_sym_#1_type}{
993       \stex_get_document_url:
994       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

995     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
996   }
997 }{
998   \str_if_empty:NF \l__stex_refs_curr_label_str {
999     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1000     \immediate\write\@auxout{
1001       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1002         \l__stex_refs_curr_label_str
1003       }
1004     }
1005   }
1006 }
1007 }

```

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

26.3 Using References

```

1008 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1009
1010 \keys_define:nn { stex / sref } {
1011   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1012   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1013   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1014   post          .tl_set:N = \l__stex_refs_post_tl ,
1015 }
1016 \cs_new_protected:Nn \__stex_refs_args:n {
1017   \tl_clear:N \l__stex_refs_linktext_tl
1018   \tl_clear:N \l__stex_refs_fallback_tl
1019   \tl_clear:N \l__stex_refs_pre_tl
1020   \tl_clear:N \l__stex_refs_post_tl
1021   \str_clear:N \l__stex_refs_repo_str
1022   \keys_set:nn { stex / sref } { #1 }
1023 }

```

The actual macro:

```

1024 \NewDocumentCommand \sref { 0{} m}{
1025   \__stex_refs_args:n { #1 }
1026   \str_if_empty:NTF \l__stex_refs_indocument_str {
1027     \str_set:Nx \l_tmpa_str { #2 }
1028     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1029     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1030       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1031         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1032           \str_clear:N \l_tmpa_str
1033         }
1034       }{
1035         \str_clear:N \l_tmpa_str
1036       }
1037     }{
1038       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1039       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1040 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1041 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1042   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1043   \str_clear:N \l_tmpa_str
1044   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1045     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1046       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1047     }{
1048       \seq_map_break:n {
1049         \str_set:Nn \l_tmpa_str { ##1 }
1050       }
1051     }
1052   }
1053 }{
1054   \str_clear:N \l_tmpa_str
1055 }
1056 }
1057 \str_if_empty:NTF \l_tmpa_str {
1058   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1059 }{
1060   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1061     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1062       \cs_if_exist:cTF{autoref}{
1063         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1064       }{
1065         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1066       }
1067     }{
1068       \ltx@ifpackageloaded{hyperref}{
1069         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1070       }{
1071         \l__stex_refs_linktext_tl
1072       }
1073     }
1074   }{
1075     \ltx@ifpackageloaded{hyperref}{
1076       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1077     }{
1078       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1079     }
1080   }
1081 }
1082 }{
1083   % TODO
1084 }
1085 }

```

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

`\srefsym`

```

1086 \NewDocumentCommand \srefsym { 0{} m}{
1087   \stex_get_symbol:n { #2 }
1088   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1089 }

```

```

1090
1091 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1092   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1093     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1094   }{
1095     \__stex_refs_args:n { #1 }
1096     \str_if_empty:NTF \l__stex_refs_indocument_str {
1097       \tl_if_exist:cTF{sref_sym_#2 _type}{
1098         % doc uri in \l_tmpb_str
1099         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1100         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1101           % reference
1102           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1103             \cs_if_exist:cTF{autoref}{
1104               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1105             }{
1106               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1107             }
1108           }{
1109             \ltx@ifpackageloaded{hyperref}{
1110               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1111             }{
1112               \l__stex_refs_linktext_tl
1113             }
1114           }
1115         }{
1116           % URL
1117           \ltx@ifpackageloaded{hyperref}{
1118             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1119           }{
1120             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1121           }
1122         }
1123       }{
1124         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1125       }
1126     }{
1127       % TODO
1128     }
1129   }
1130 }

```

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

\srefsymuri

```

1131 \cs_new_protected:Npn \srefsymuri #1 #2 {
1132   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1133 }

```

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

```

1134 </package>

```

Chapter 27

STEX -Modules Implementation

```
1135 <*package>
1136
1137 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1138
1139 <@@=stex_modules>
1140
1141   Warnings and error messages
1142   \msg_new:nnn{stex}{error/unknownmodule}{
1143     No~module~#1~found
1144   }
1145   \msg_new:nnn{stex}{error/syntax}{
1146     Syntax~error:~#1
1147   }
1148   \msg_new:nnn{stex}{error/siglanguage}{
1149     Module~#1~declares~signature~#2,~but~does~not~
1150     declare~its~language
1151   }
1152   \msg_new:nnn{stex}{warning/deprecated}{
1153     #1~is~deprecated;~please~use~#2~instead!
1154   }
1155   \msg_new:nnn{stex}{error/conflictingmodules}{
1156     Conflicting~imports~for~module~#1
1157   }
```

`\l_stex_current_module_str` The current module:

```
1157 \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

```
1158 \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
1159 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1160   \str_if_empty:NTF \l_stex_current_module_str
1161   \prg_return_false: \prg_return_true:
1162 }

(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
1163 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1164   \prop_if_exist:cTF { c_stex_module_#1_prop }
1165   \prg_return_true: \prg_return_false:
1166 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1167 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1168   \stex_add_to_current_module:n { #1 }
1169   \stex_do_up_to_module:n { #1 }
1170 }}
1171 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1172
1173 \cs_new_protected:Nn \stex_add_to_current_module:n {
1174   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1175 }
1176 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1177 \cs_new_protected:Npn \STEXexport {
1178   \beginngroup
1179   \newlinechar=-1\relax
1180   \endlinechar=-1\relax
1181   %\catcode'\ = 9\relax
1182   \expandafter\endgroup\__stex_modules_export:n
1183 }
1184 \cs_new_protected:Nn \__stex_modules_export:n {
1185   \ignorespaces #1
1186   \stex_add_to_current_module:n { \ignorespaces #1 }
1187   \stex_smsmode_do:
1188 }
1189 \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
1190 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1191   \str_set:Nx \l_tmpa_str { #1 }
1192   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1193 }

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

```

`\stex_add_import_to_current_module:n`

```

1194 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1195   \str_set:Nx \l_tmpa_str { #1 }
1196   \exp_args:Nno
1197   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1198     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1199   }
1200 }

```

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

`\stex_collect_imports:n`

```

1201 \cs_new_protected:Nn \stex_collect_imports:n {
1202   \seq_clear:N \l_stex_collect_imports_seq
1203   \__stex_modules_collect_imports:n {#1}
1204 }
1205 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1206   \seq_map_inline:cn {c_stex_module_#1_imports} {
1207     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1208       \__stex_modules_collect_imports:n { ##1 }
1209     }
1210   }
1211   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1212     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1213   }
1214 }

```

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

`\stex_do_up_to_module:n`

```

1215 \int_new:N \l__stex_modules_group_depth_int
1216 \cs_new_protected:Nn \stex_do_up_to_module:n {
1217   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1218     #1
1219   }{
1220     #1
1221     \expandafter \tl_gset:Nn
1222     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1223     \expandafter\expandafter\expandafter\endcsname
1224     \expandafter\expandafter\expandafter { \csname
1225       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1226     \aftergroup\__stex_modules_aftergroup_do:
1227   }
1228 }
1229 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1230 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1231   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1232     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1233   }}
1234   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1235     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1236     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1237   }{
1238     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```



```

1239     \aftergroup\__stex_modules_aftergroup_do:
1240   }
1241 }
1242 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1243   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1244 }

```

(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).

```

1245

```

(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.

```

1246 \str_new:N \l_stex_module_ns_str
1247 \str_new:N \l_stex_module_subpath_str
1248 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1249   \seq_set_eq:NN \l_tmpa_seq #2
1250   % split off file extension
1251   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1252   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1253   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1254   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1255
1256   \bool_set_true:N \l_tmpa_bool
1257   \bool_while_do:Nn \l_tmpa_bool {
1258     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1259     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1260       {source} { \bool_set_false:N \l_tmpa_bool }
1261     }{}{
1262       \seq_if_empty:NT \l_tmpa_seq {
1263         \bool_set_false:N \l_tmpa_bool
1264       }
1265     }
1266   }
1267
1268   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1269   % \l_tmpa_seq <- sub-path relative to archive
1270   \str_if_empty:NTF \l_stex_module_subpath_str {
1271     \str_set:Nx \l_stex_module_ns_str {#1}
1272   }{
1273     \str_set:Nx \l_stex_module_ns_str {
1274       #1/\l_stex_module_subpath_str
1275     }
1276   }
1277 }
1278
1279 \cs_new_protected:Nn \stex_modules_current_namespace: {
1280   \str_clear:N \l_stex_module_subpath_str
1281   \prop_if_exist:NTF \l_stex_current_repository_prop {
1282     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1283     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1284   }{
1285     % split off file extension
1286     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1287     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1288     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1289     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1290     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1291     \str_set:Nx \l_stex_module_ns_str {
1292       file:/\stex_path_to_string:N \l_tmpa_seq
1293     }
1294   }
1295 }

```

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

27.1 The smodule environment

smodule arguments:

```

1296 \keys_define:nn { stex / module } {
1297   title      .tl_set:N      = \smodulename ,
1298   type       .str_set_x:N   = \smodulename ,
1299   id         .str_set_x:N   = \smoduleid ,
1300   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1301   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1302   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1303   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1304   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1305   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1306   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1307   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1308 }
1309
1310 \cs_new_protected:Nn \__stex_modules_args:n {
1311   \str_clear:N \smodulename
1312   \str_clear:N \smodulename
1313   \str_clear:N \smoduleid
1314   \str_clear:N \l_stex_module_ns_str
1315   \str_clear:N \l_stex_module_deprecate_str
1316   \str_clear:N \l_stex_module_lang_str
1317   \str_clear:N \l_stex_module_sig_str
1318   \str_clear:N \l_stex_module_creators_str
1319   \str_clear:N \l_stex_module_contributors_str
1320   \str_clear:N \l_stex_module_meta_str
1321   \str_clear:N \l_stex_module_srccite_str
1322   \keys_set:nn { stex / module } { #1 }
1323 }
1324
1325 % module parameters here? In the body?
1326

```

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

```

1327 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1328 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1329 \str_set:Nx \l_stex_module_name_str { #2 }
1330 \__stex_modules_args:n { #1 }

```

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

```

1331 \stex_if_in_module:TF {
1332   % Nested module
1333   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1334   { ns } \l_stex_module_ns_str
1335   \str_set:Nx \l_stex_module_name_str {
1336     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1337     { name } / \l_stex_module_name_str
1338   }
1339   \str_if_empty:NT \l_stex_module_lang_str {
1340     \str_set:Nx \l_stex_module_lang_str {
1341       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1342       { lang }
1343     }
1344   }
1345 }{
1346   % not nested:
1347   \str_if_empty:NT \l_stex_module_ns_str {
1348     \stex_modules_current_namespace:
1349     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1350       / {\l_stex_module_ns_str}
1351     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1352     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1353       \str_set:Nx \l_stex_module_ns_str {
1354         \stex_path_to_string:N \l_tmpa_seq
1355       }
1356     }
1357   }
1358 }

```

Next, we determine the language of the module:

```

1359 \str_if_empty:NT \l_stex_module_lang_str {
1360   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1361   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1362   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1363   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1364     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1365       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1366     }
1367   }
1368   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1369   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1370     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1371     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1372       inferred~from~file~name}
1373   }
1374 }
1375
1376 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1377 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1378 \l_tmpa_str {
1379 \ltx@ifpackageloaded{babel}{
1380 \exp_args:Nx \selectlanguage { \l_tmpa_str }
1381 }{}
1382 } {
1383 \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1384 }
1385 }}

```

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

```

1386 \str_if_empty:NTF \l_stex_module_sig_str {
1387 \exp_args:Nnx \prop_gset_from_keyval:cn {
1388 c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1389 } {
1390 name = \l_stex_module_name_str ,
1391 ns = \l_stex_module_ns_str ,
1392 file = \exp_not:o { \g_stex_currentfile_seq } ,
1393 lang = \l_stex_module_lang_str ,
1394 sig = \l_stex_module_sig_str ,
1395 deprecate = \l_stex_module_deprecate_str ,
1396 meta = \l_stex_module_meta_str
1397 }
1398 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1399 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1400 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1401 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1402 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1403 \str_if_empty:NT \l_stex_module_meta_str {
1404 \str_set:Nx \l_stex_module_meta_str {
1405 \c_stex_metatheory_ns_str ? Metatheory
1406 }
1407 }
1408 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1409 \bool_set_true:N \l_stex_in_meta_bool
1410 \exp_args:Nx \stex_add_to_current_module:n {
1411 \bool_set_true:N \l_stex_in_meta_bool
1412 \stex_activate_module:n {\l_stex_module_meta_str}
1413 \bool_set_false:N \l_stex_in_meta_bool
1414 }
1415 \stex_activate_module:n {\l_stex_module_meta_str}
1416 \bool_set_false:N \l_stex_in_meta_bool
1417 }
1418 }{
1419 \str_if_empty:NT \l_stex_module_lang_str {
1420 \msg_error:nnxx{stex}{error/siglanguage}{
1421 \l_stex_module_ns_str?\l_stex_module_name_str
1422 }\l_stex_module_sig_str}
1423 }
1424 \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1425 \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{

```

```

1426     \stex_debug:nn{modules}{(already exists)}
1427   }{
1428     \stex_debug:nn{modules}{(needs loading)}
1429     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1430     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1431     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1432     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1433     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1434     \str_set:Nx \l_tmpa_str {
1435       \stex_path_to_string:N \l_tmpa_seq /
1436       \l_tmpa_str . \l_stex_module_sig_str .tex
1437     }
1438     \IfFileExists \l_tmpa_str {
1439       \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1440         \str_clear:N \l_stex_current_module_str
1441         \seq_clear:N \l_stex_all_modules_seq
1442         \stex_debug:nn{modules}{Loading~signature}
1443       }
1444     }{
1445       \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1446     }
1447   }
1448   \stex_if_smsmode:F {
1449     \stex_activate_module:n {
1450       \l_stex_module_ns_str ? \l_stex_module_name_str
1451     }
1452   }
1453   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1454 }
1455 \str_if_empty:NF \l_stex_module_deprecate_str {
1456   \msg_warning:nnxx{stex}{warning/deprecated}{
1457     Module~\l_stex_current_module_str
1458   }{
1459     \l_stex_module_deprecate_str
1460   }
1461 }
1462 \seq_put_right:Nx \l_stex_all_modules_seq {
1463   \l_stex_module_ns_str ? \l_stex_module_name_str
1464 }
1465 \tl_clear:c{l__stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1466 }

```

(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}`

```

1467 \cs_new_protected:Nn \_stex_modules_begin_module: {
1468   \stex_reactivate_macro:N \STEXexport
1469   \stex_reactivate_macro:N \importmodule
1470   \stex_reactivate_macro:N \symdecl
1471   \stex_reactivate_macro:N \notation
1472   \stex_reactivate_macro:N \symdef
1473 }

```

```

1474 \stex_debug:nn{modules}{
1475   New~module:\
1476   Namespace:~\l_stex_module_ns_str\
1477   Name:~\l_stex_module_name_str\
1478   Language:~\l_stex_module_lang_str\
1479   Signature:~\l_stex_module_sig_str\
1480   Metatheory:~\l_stex_module_meta_str\
1481   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1482 }
1483
1484 \stex_if_do_html:T{
1485   \begin{stex_annotate_env} {theory} {
1486     \l_stex_module_ns_str ? \l_stex_module_name_str
1487   }
1488
1489   \stex_annotate_invisible:nnn{header}{} {
1490     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1491     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1492     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1493       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1494     }
1495     \str_if_empty:NF \smodulotype {
1496       \stex_annotate:nnn{type}{\smodulotype}{}
1497     }
1498   }
1499 }
1500 % TODO: Inherit metatheory for nested modules?
1501 }
1502 \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}

```

1503 \cs_new_protected:Nn \_stex_modules_end_module: {
1504   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1505   \_stex_reset_up_to_module:n \l_stex_current_module_str
1506   \stex_if_smsmode:T {
1507     \stex_persist:x {
1508       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1509         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1510       }
1511       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1512         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1513       }
1514       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1515         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1516       }
1517       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1518     }
1519     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1520     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1521   }
1522 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1523 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1524 \NewDocumentEnvironment { smodule } { 0{} m } {
1525   \stex_module_setup:nn{#1}{#2}
1526   \par
1527   \stex_if_smsmode:F{
1528     \tl_if_empty:NF \smodulename {
1529       \exp_args:No \stex_document_title:n \smodulename
1530     }
1531     \tl_clear:N \l_tmpa_tl
1532     \clist_map_inline:Nn \smodulename {
1533       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1534         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1535       }
1536     }
1537     \tl_if_empty:NTF \l_tmpa_tl {
1538       \__stex_modules_smodule_start:
1539     }{
1540       \l_tmpa_tl
1541     }
1542   }
1543   \__stex_modules_begin_module:
1544   \str_if_empty:NF \smoduleid {
1545     \stex_ref_new_doc_target:n \smoduleid
1546   }
1547   \stex_smsmode_do:
1548 } {
1549   \__stex_modules_end_module:
1550   \stex_if_smsmode:F {
1551     \end{stex_annotate_env}
1552     \clist_set:No \l_tmpa_clist \smodulename
1553     \tl_clear:N \l_tmpa_tl
1554     \clist_map_inline:Nn \l_tmpa_clist {
1555       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1556         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1557       }
1558     }
1559     \tl_if_empty:NTF \l_tmpa_tl {
1560       \__stex_modules_smodule_end:
1561     }{
1562       \l_tmpa_tl
1563     }
1564   }
1565 }

```

\stexpatchmodule

```

1566 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1567 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1568
1569 \newcommand\stexpatchmodule[3] [] {
1570   \str_set:Nx \l_tmpa_str{ #1 }
1571   \str_if_empty:NTF \l_tmpa_str {

```

```

1572     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1573     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1574   }{
1575     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1576     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1577   }
1578 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1579 \NewDocumentCommand \STEXModule { m } {
1580   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1581   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1582   \tl_set:Nn \l_tmpa_tl {
1583     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1584   }
1585   \seq_map_inline:Nn \l_stex_all_modules_seq {
1586     \str_set:Nn \l_tmpb_str { ##1 }
1587     \str_if_eq:eeT { \l_tmpa_str } {
1588       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1589     } {
1590       \seq_map_break:n {
1591         \tl_set:Nn \l_tmpa_tl {
1592           \stex_invoke_module:n { ##1 }
1593         }
1594       }
1595     }
1596   }
1597   \l_tmpa_tl
1598 }
1599
1600 \cs_new_protected:Nn \stex_invoke_module:n {
1601   \stex_debug:nn{modules}{Invoking~module~#1}
1602   \peek_charcode_remove:NTF ! {
1603     \__stex_modules_invoke_uri:nN { #1 }
1604   } {
1605     \peek_charcode_remove:NTF ? {
1606       \__stex_modules_invoke_symbol:nn { #1 }
1607     } {
1608       \msg_error:nnx{stex}{error/syntax}{
1609         ?~or~!~expected~after~
1610         \c_backslash_str STEXModule{#1}
1611       }
1612     }
1613   }
1614 }
1615
1616 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1617   \str_set:Nn #2 { #1 }
1618 }

```



```

1619
1620 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1621   \stex_invoke_symbol:n{#1?#2}
1622 }

```

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

\stex_activate_module:n

```

1623 \bool_new:N \l_stex_in_meta_bool
1624 \bool_set_false:N \l_stex_in_meta_bool
1625 \cs_new_protected:Nn \stex_activate_module:n {
1626   \stex_debug:nn{modules}{Activating~module~#1}
1627   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1628     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1629     \use:c{ c_stex_module_#1_code }
1630   }
1631 }

```

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

```

1632 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1633 <*package>
1634
1635 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1636
```

28.1 SMS Mode

```
1637 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1638 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1639 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1640 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1641
1642 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1643   \makeatletter
1644   \makeatother
1645   \ExplSyntaxOn
1646   \ExplSyntaxOff
1647   \rustexBREAK
1648 }
1649
1650 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1651   \symdef
1652   \importmodule
1653   \notation
1654   \symdecl
1655   \STEXexport
1656   \inlineass
1657   \inlinedef
1658   \inlineex
1659   \endinput
1660   \setnotation
```

```

1661 \copynotation
1662 \assign
1663 \renamedekl
1664 \donotcopy
1665 \instantiate
1666 }
1667
1668 \exp_args:N Nx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1669   \tl_to_str:n {
1670     smodule,
1671     copymodule,
1672     interpretmodule,
1673     sdefinition,
1674     sexample,
1675     sassertion,
1676     sparagraph,
1677     mathstructure
1678   }
1679 }

```

(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`

```

1680 \bool_new:N \g__stex_smsmode_bool
1681 \bool_set_false:N \g__stex_smsmode_bool
1682 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1683   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1684 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1685 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1686   \vbox_set:Nn \l_tmpa_box {
1687     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1688     \bool_gset_true:N \g__stex_smsmode_bool
1689     #2
1690     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1691   }
1692   \box_clear:N \l_tmpa_box
1693 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1694 \quark_new:N \q__stex_smsmode_break
1695
1696 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1697   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1698   \stex_smsmode_do:
1699 }
1700
1701 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1702   \_stex_modules_args:n{#1}

```

```

1703 \stex_if_in_module:F {
1704   \str_if_empty:NF \l_stex_module_sig_str {
1705     \stex_modules_current_namespace:
1706     \str_set:Nx \l_stex_module_name_str { #2 }
1707     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1708       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1709       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1710       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1711       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1712       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1713       \str_set:Nx \l_tmpa_str {
1714         \stex_path_to_string:N \l_tmpa_seq /
1715         \l_tmpa_str . \l_stex_module_sig_str .tex
1716       }
1717       \IfFileExists \l_tmpa_str {
1718         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1719       }{
1720         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1721       }
1722     }
1723   }
1724 }
1725 }
1726
1727 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1728   \stex_filestack_push:n{#1}
1729   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1730   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1731   % ----- new -----
1732   \__stex_smsmode_in_smsmode:nn{#1}{
1733     \let\importmodule\__stex_smsmode_importmodule:
1734     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1735     \let\__stex_modules_begin_module:\relax
1736     \let\__stex_modules_end_module:\relax
1737     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1738     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1739     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1740     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1741     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1742     \everyeof{\q__stex_smsmode_break\noexpand}
1743     \expandafter\expandafter\expandafter
1744     \stex_smsmode_do:
1745     \csname @ @ input\endcsname "#1"\relax
1746
1747     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1748       \stex_filestack_push:n{##1}
1749       \expandafter\expandafter\expandafter
1750       \stex_smsmode_do:
1751       \csname @ @ input\endcsname "##1"\relax
1752       \stex_filestack_pop:
1753     }
1754   }
1755   % ----- new -----
1756   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1757 #2
1758 % ----- new -----
1759 \begingroup
1760 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1761 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1762   \stex_import_module_uri:nn ##1
1763   \stex_import_require_module:nnnn
1764   \l_stex_import_ns_str
1765   \l_stex_import_archive_str
1766   \l_stex_import_path_str
1767   \l_stex_import_name_str
1768 }
1769 \endgroup
1770 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1771 % ----- new -----
1772 \everyeof{\q__stex_smsmode_break\noexpand}
1773 \expandafter\expandafter\expandafter
1774 \stex_smsmode_do:
1775 \csname @ @ input\endcsname "#1"\relax
1776 }
1777 \stex_filestack_pop:
1778 }

```

(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:

```

1779 \cs_new_protected:Npn \stex_smsmode_do: {
1780   \stex_if_smsmode:T {
1781     \__stex_smsmode_do:w
1782   }
1783 }
1784 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1785   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1786     \expandafter\if\expandafter\relax\noexpand#1
1787     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1788     \else\expandafter\__stex_smsmode_do:w\fi
1789   }{
1790     \__stex_smsmode_do:w %#1
1791   }
1792 }
1793 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1794   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1795     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1796       #1\__stex_smsmode_do:w
1797     }{
1798       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1799         #1
1800       }{
1801         \cs_if_eq:NNTF \begin #1 {
1802           \__stex_smsmode_check_begin:n
1803         }{
1804           \cs_if_eq:NNTF \end #1 {
1805             \__stex_smsmode_check_end:n

```

```

1806         }{
1807         \__stex_smsmode_do:w
1808         }
1809     }
1810 }
1811 }
1812 }
1813 }
1814
1815 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1816 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1817 \begin{#1}
1818 }{
1819 \__stex_smsmode_do:w
1820 }
1821 }
1822 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1823 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1824 \end{#1}\__stex_smsmode_do:w
1825 }{
1826 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1827 }
1828 }

```

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

28.2 Inheritance

```

1829 <@@=stex_importmodule>

\stex_import_module_uri:nn

1830 \cs_new_protected:Nn \stex_import_module_uri:nn {
1831 \str_set:Nx \l_stex_import_archive_str { #1 }
1832 \str_set:Nn \l_stex_import_path_str { #2 }
1833
1834 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1835 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1836 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1837
1838 \stex_modules_current_namespace:
1839 \bool_lazy_all:nTF {
1840 {\str_if_empty_p:N \l_stex_import_archive_str}
1841 {\str_if_empty_p:N \l_stex_import_path_str}
1842 {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1843 }{
1844 \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1845 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1846 }{
1847 \str_if_empty:NT \l_stex_import_archive_str {
1848 \prop_if_exist:NT \l_stex_current_repository_prop {
1849 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1850 }
1851 }
1852 \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1853     \str_if_empty:NF \l_stex_import_path_str {
1854         \str_set:Nx \l_stex_import_ns_str {
1855             \l_stex_module_ns_str / \l_stex_import_path_str
1856         }
1857     }
1858 }{
1859     \stex_require_repository:n \l_stex_import_archive_str
1860     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1861     \l_stex_import_ns_str
1862     \str_if_empty:NF \l_stex_import_path_str {
1863         \str_set:Nx \l_stex_import_ns_str {
1864             \l_stex_import_ns_str / \l_stex_import_path_str
1865         }
1866     }
1867 }
1868 }
1869 }

```

(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 1870 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str 1871 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str 1872 \str_new:N \l_stex_import_path_str
1873 \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:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1874 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1875     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1876
1877         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1878
1879         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/{}} {#4}
1880         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1881
1882         %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
1883
1884         % archive
1885         \str_set:Nx \l_tmpa_str { #2 }
1886         \str_if_empty:NTF \l_tmpa_str {
1887             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1888         } {
1889             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1890             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1891             \seq_put_right:Nn \l_tmpa_seq { source }
1892         }
1893
1894         % path
1895         \str_set:Nx \l_tmpb_str { #3 }
1896         \str_if_empty:NTF \l_tmpb_str {
1897             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1898

```

```

1899 \ltx@ifpackageloaded{babel} {
1900   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1901     { \language } \l_tmpb_str {
1902       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1903     }
1904   } {
1905     \str_clear:N \l_tmpb_str
1906   }
1907
1908   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1909   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1910     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1911   }{
1912     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1913     \IfFileExists{ \l_tmpa_str.tex }{
1914       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1915     }{
1916       % try english as default
1917       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1918       \IfFileExists{ \l_tmpa_str.en.tex }{
1919         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1920       }{
1921         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1922       }
1923     }
1924   }
1925
1926 } {
1927   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1928   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1929
1930   \ltx@ifpackageloaded{babel} {
1931     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1932       { \language } \l_tmpb_str {
1933       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1934     }
1935   } {
1936     \str_clear:N \l_tmpb_str
1937   }
1938
1939   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1940
1941   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1942   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1943     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1944   }{
1945     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1946     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1947       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1948     }{
1949       % try english as default
1950       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1951       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1952         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```



```

1953     }{
1954     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1955     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1956         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1957     }{
1958         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1959         \IfFileExists{ \l_tmpa_str.tex }{
1960             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1961         }{
1962             % try english as default
1963             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1964             \IfFileExists{ \l_tmpa_str.en.tex }{
1965                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1966             }{
1967                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1968             }
1969         }
1970     }
1971 }
1972 }
1973 }
1974 }
1975
1976 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1977     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1978         \seq_clear:N \l_stex_all_modules_seq
1979         \str_clear:N \l_stex_current_module_str
1980         \str_set:Nx \l_tmpb_str { #2 }
1981         \str_if_empty:NF \l_tmpb_str {
1982             \stex_set_current_repository:n { #2 }
1983         }
1984         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1985     }
1986
1987     \stex_if_module_exists:nF { #1 ? #4 } {
1988         \msg_error:nnx{stex}{error/unknownmodule}{
1989             #1?#4~(in~file~\g__stex_importmodule_file_str)
1990         }
1991     }
1992 }
1993
1994 }
1995 \stex_activate_module:n { #1 ? #4 }
1996 }

```

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

`\importmodule`

```

1997 \NewDocumentCommand \importmodule { 0{ } m } {
1998     \stex_import_module_uri:nn { #1 } { #2 }
1999     \stex_debug:nn{modules}{Importing~module:~
2000         \l_stex_import_ns_str ? \l_stex_import_name_str
2001     }
2002     \stex_import_require_module:nnnn

```

```

2003 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2004 { \l_stex_import_path_str } { \l_stex_import_name_str }
2005 \stex_if_smsmode:F {
2006   \stex_annotate_invisible:nnn
2007   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2008 }
2009 \exp_args:Nx \stex_add_to_current_module:n {
2010   \stex_import_require_module:nnnn
2011   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2012   { \l_stex_import_path_str } { \l_stex_import_name_str }
2013 }
2014 \exp_args:Nx \stex_add_import_to_current_module:n {
2015   \l_stex_import_ns_str ? \l_stex_import_name_str
2016 }
2017 \stex_smsmode_do:
2018 \ignorespacesandpars
2019 }
2020 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2021 \NewDocumentCommand \usemodule { 0{} m } {
2022   \stex_if_smsmode:F {
2023     \stex_import_module_uri:nn { #1 } { #2 }
2024     \stex_import_require_module:nnnn
2025     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2026     { \l_stex_import_path_str } { \l_stex_import_name_str }
2027     \stex_annotate_invisible:nnn
2028     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2029   }
2030   \stex_smsmode_do:
2031   \ignorespacesandpars
2032 }

```

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

```

2033 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2034   \tl_if_empty:nF{#2}{
2035     \clist_set:Nn \l_tmpa_clist {#2}
2036     \clist_map_inline:Nn \l_tmpa_clist {
2037       \tl_if_head_eq_charcode:nNTF {##1}[{
2038         #1 ##1
2039       }{
2040         #1{##1}
2041       }
2042     }
2043   }
2044 }
2045 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2046
2047
2048 </package>

```

Chapter 29

STEX -Symbols Implementation

```
2049 <*package>
2050
2051 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2052
2053 Warnings and error messages
2054 \msg_new:nnn{stex}{error/wrongargs}{
2055   args~value~in~symbol~declaration~for~#1~
2056   needs~to~be~i,~a,~b~or~B,~but~#2~given
2057 }
2058 \msg_new:nnn{stex}{error/unknownsymbol}{
2059   No~symbol~#1~found!
2060 }
2061 \msg_new:nnn{stex}{error/seqlength}{
2062   Expected~#1~arguments;~got~#2!
2063 }
2064 \msg_new:nnn{stex}{error/unknownnotation}{
2065   Unknown~notation~#1~for~#2!
2066 }
```

29.1 Symbol Declarations

```
2066 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2067 \cs_new_protected:Nn \stex_all_symbols:n {
2068   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2069   \seq_map_inline:Nn \l_stex_all_modules_seq {
2070     \seq_map_inline:cn{c_stex_module_##1_constants}{
2071       \__stex_symdecl_all_symbols_cs{##1?####1}
2072     }
2073   }
2074 }
```

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

`\STEXsymbol`

```
2075 \NewDocumentCommand \STEXsymbol { m } {
2076   \stex_get_symbol:n { #1 }
2077   \exp_args:No
2078   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2079 }
```

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

`symdecl` arguments:

```
2080 \keys_define:nn { stex / symdecl } {
2081   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2082   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2083   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2084   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2085   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2086   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2087   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2088   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2089   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2090   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2091   assoc     .choices:nn =
2092     {bin,binl,binr,pre,conj,pwconj}
2093     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2094 }
2095
2096 \bool_new:N \l_stex_symdecl_make_macro_bool
2097
2098 \cs_new_protected:Nn \__stex_symdecl_args:n {
2099   \str_clear:N \l_stex_symdecl_name_str
2100   \str_clear:N \l_stex_symdecl_args_str
2101   \str_clear:N \l_stex_symdecl_deprecate_str
2102   \str_clear:N \l_stex_symdecl_reorder_str
2103   \str_clear:N \l_stex_symdecl_assoctype_str
2104   \bool_set_false:N \l_stex_symdecl_local_bool
2105   \tl_clear:N \l_stex_symdecl_type_tl
2106   \tl_clear:N \l_stex_symdecl_definiens_tl
2107
2108   \keys_set:nn { stex / symdecl } { #1 }
2109 }
```

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

```
2110
2111 \NewDocumentCommand \symdecl { s m O{} } {
2112   \__stex_symdecl_args:n { #3 }
2113   \IfBooleanTF #1 {
2114     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2115   } {
2116     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2117   }
2118   \stex_symdecl_do:n { #2 }
2119   \stex_smsmode_do:
2120 }
```

```

2121
2122 \cs_new_protected:Nn \stex_symdecl_do:nn {
2123   \__stex_symdecl_args:n{#1}
2124   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2125   \stex_symdecl_do:n{#2}
2126 }
2127
2128 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2129 \cs_new_protected:Nn \stex_symdecl_do:n {
2130   \stex_if_in_module:F {
2131     % TODO throw error? some default namespace?
2132   }
2133
2134   \str_if_empty:NT \l_stex_symdecl_name_str {
2135     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2136   }
2137
2138   \prop_if_exist:cT { l_stex_symdecl_
2139     \l_stex_current_module_str ?
2140     \l_stex_symdecl_name_str
2141     _prop
2142   }{
2143     % TODO throw error (beware of circular dependencies)
2144   }
2145
2146   \prop_clear:N \l_tmpa_prop
2147   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2148   \seq_clear:N \l_tmpa_seq
2149   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2150   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2151
2152   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2153     \str_if_empty:NF \l_stex_module_deprecate_str {
2154       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2155     }
2156   }
2157   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2158
2159   \exp_args:No \stex_add_constant_to_current_module:n {
2160     \l_stex_symdecl_name_str
2161   }
2162
2163   % arity/args
2164   \int_zero:N \l_tmpb_int
2165
2166   \bool_set_true:N \l_tmpa_bool
2167   \str_map_inline:Nn \l_stex_symdecl_args_str {
2168     \token_case_meaning:NnF ##1 {
2169       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2170       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2171     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2172     {\tl_to_str:n a} {
2173       \bool_set_false:N \l_tmpa_bool
2174       \int_incr:N \l_tmpb_int
2175     }
2176     {\tl_to_str:n B} {
2177       \bool_set_false:N \l_tmpa_bool
2178       \int_incr:N \l_tmpb_int
2179     }
2180   }{
2181     \msg_error:nnxx{stex}{error/wrongargs}{
2182       \l_stex_current_module_str ?
2183       \l_stex_symdecl_name_str
2184     }{##1}
2185   }
2186 }
2187 \bool_if:NTF \l_tmpa_bool {
2188   % possibly numeric
2189   \str_if_empty:NTF \l_stex_symdecl_args_str {
2190     \prop_put:Nnn \l_tmpa_prop { args } {}
2191     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2192   }{
2193     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2194     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2195     \str_clear:N \l_tmpa_str
2196     \int_step_inline:nn \l_tmpa_int {
2197       \str_put_right:Nn \l_tmpa_str i
2198     }
2199     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2200   }
2201 } {
2202   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2203   \prop_put:Nnx \l_tmpa_prop { arity }
2204   { \str_count:N \l_stex_symdecl_args_str }
2205 }
2206 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2207
2208 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2209   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2210 }{
2211   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2212 }
2213
2214 % semantic macro
2215
2216 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2217   \exp_args:Nx \stex_do_up_to_module:n {
2218     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2219       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2220     }}
2221   }
2222 }
2223
2224 \stex_debug:nn{symbols}{New~symbol:~

```

```

2225 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2226 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2227 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2228 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2229 }
2230
2231 % circular dependencies require this:
2232 \stex_if_do_html:T {
2233   \stex_annotate_invisible:nnn {symdecl} {
2234     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2235   } {
2236     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2237       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2238     }
2239     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$}
2240     \prop_item:Nn \l_tmpa_prop { args }
2241   }
2242   \stex_annotate_invisible:nnn{macroname}{#1}{}
2243   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2244     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2245   }
2246   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2247     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2248   }
2249   \str_if_empty:NF \l_stex_symdecl_reorder_str {
2250     \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2251   }
2252 }
2253 }
2254 }
2255 \prop_if_exist:cF {
2256   \l_stex_symdecl_
2257   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2258   _prop
2259 } {
2260   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2261     \__stex_symdecl_restore_symbol:nnnnnnn
2262     {\l_stex_symdecl_name_str}
2263     { \prop_item:Nn \l_tmpa_prop {args} }
2264     { \prop_item:Nn \l_tmpa_prop {arity} }
2265     { \prop_item:Nn \l_tmpa_prop {assocs} }
2266     { \prop_item:Nn \l_tmpa_prop {defined} }
2267     {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2268     {\l_stex_current_module_str}
2269 }
2270 }
2271 }
2272 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2273   \prop_clear:N \l_tmpa_prop
2274   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2275   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2276   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2277   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2278   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }

```

```

2279 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2280 \tl_if_empty:nF{#6}{
2281   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2282 }
2283 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2284 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2285 }

```

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

`\stex_get_symbol:n`

```

2286 \str_new:N \l_stex_get_symbol_uri_str
2287
2288 \cs_new_protected:Nn \stex_get_symbol:n {
2289   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2290     \tl_set:Nn \l_tmpa_tl { #1 }
2291     \__stex_symdecl_get_symbol_from_cs:
2292   }{
2293     % argument is a string
2294     % is it a command name?
2295     \cs_if_exist:cTF { #1 }{
2296       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2297       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2298       \str_if_empty:NTF \l_tmpa_str {
2299         \exp_args:Nx \cs_if_eq:NNTF {
2300           \tl_head:N \l_tmpa_tl
2301         } \stex_invoke_symbol:n {
2302           \__stex_symdecl_get_symbol_from_cs:
2303         }{
2304           \__stex_symdecl_get_symbol_from_string:n { #1 }
2305         }
2306       } {
2307         \__stex_symdecl_get_symbol_from_string:n { #1 }
2308       }
2309     }{
2310       % argument is not a command name
2311       \__stex_symdecl_get_symbol_from_string:n { #1 }
2312       % \l_stex_all_symbols_seq
2313     }
2314   }
2315   \str_if_eq:eeF {
2316     \prop_item:cn {
2317       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2318     }{ deprecate }
2319   }{
2320     \msg_warning:nxxx{stex}{warning/deprecated}{
2321       Symbol~\l_stex_get_symbol_uri_str
2322     }{
2323       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2324     }
2325   }
2326 }
2327
2328 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```



```

2329 \tl_set:Nn \l_tmpa_tl {
2330   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2331 }
2332 \str_set:Nn \l_tmpa_str { #1 }
2333
2334 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2335
2336 \str_if_in:NnTF \l_tmpa_str ? {
2337   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2338   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2339   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2340 }{
2341   \str_clear:N \l_tmpb_str
2342 }
2343 \str_if_empty:NNTF \l_tmpb_str {
2344   \seq_map_inline:Nn \l_stex_all_modules_seq {
2345     \seq_map_inline:cn{c_stex_module_###1_constants}{
2346       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2347         \seq_map_break:n{\seq_map_break:n{
2348           \tl_set:Nn \l_tmpa_tl {
2349             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2350           }
2351         }}
2352       }
2353     }
2354   }
2355 }{
2356   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2357   \seq_map_inline:Nn \l_stex_all_modules_seq {
2358     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2359       \seq_map_inline:cn{c_stex_module_###1_constants}{
2360         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2361           \seq_map_break:n{\seq_map_break:n{
2362             \tl_set:Nn \l_tmpa_tl {
2363               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2364             }
2365           }}
2366         }
2367       }
2368     }
2369   }
2370 }
2371
2372 \l_tmpa_tl
2373 }
2374
2375 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2376   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2377   { \tl_tail:N \l_tmpa_tl }
2378   \tl_if_single:NNTF \l_tmpa_tl {
2379     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2380       \exp_after:wN \str_set:Nn \exp_after:wN
2381       \l_stex_get_symbol_uri_str \l_tmpa_tl
2382     }{

```

```

2383     % TODO
2384     % tail is not a single group
2385   }
2386 }{
2387   % TODO
2388   % tail is not a single group
2389 }
2390 }

```

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

29.2 Notations

```

2391 <@@=stex_notation>

notation arguments:
2392 \keys_define:nn { stex / notation } {
2393   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2394   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2395   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2396   op       .tl_set:N = \l__stex_notation_op_tl ,
2397   primary .bool_set:N = \l__stex_notation_primary_bool ,
2398   primary .default:n = {true} ,
2399   unknown .code:n = \str_set:Nx
2400             \l__stex_notation_variant_str \l_keys_key_str
2401 }
2402
2403 \cs_new_protected:Nn \stex_notation_args:n {
2404   % \str_clear:N \l__stex_notation_lang_str
2405   \str_clear:N \l__stex_notation_variant_str
2406   \str_clear:N \l__stex_notation_prec_str
2407   \tl_clear:N \l__stex_notation_op_tl
2408   \bool_set_false:N \l__stex_notation_primary_bool
2409
2410   \keys_set:nn { stex / notation } { #1 }
2411 }

\notation
2412 \NewDocumentCommand \notation { s m 0{}} {
2413   \stex_notation_args:n { #3 }
2414   \tl_clear:N \l_stex_symdecl_definiens_tl
2415   \stex_get_symbol:n { #2 }
2416   \tl_set:Nn \l_stex_notation_after_do_tl {
2417     \__stex_notation_final:
2418     \IfBooleanTF#1{
2419       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2420     }{}
2421     \stex_smsmode_do:\ignorespacesandpars
2422   }
2423   \stex_notation_do:nnnnn
2424   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2425   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2426   { \l__stex_notation_variant_str }
2427   { \l__stex_notation_prec_str }

```

```

2428 }
2429 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2430 \seq_new:N \l__stex_notation_precedences_seq
2431 \tl_new:N \l__stex_notation_opprec_tl
2432 \int_new:N \l__stex_notation_currarg_int
2433 \tl_new:N \stex_symbol_after_invokation_tl
2434
2435 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2436   \let\l_stex_current_symbol_str\relax
2437   \seq_clear:N \l__stex_notation_precedences_seq
2438   \tl_clear:N \l__stex_notation_opprec_tl
2439   \str_set:Nx \l__stex_notation_args_str { #1 }
2440   \str_set:Nx \l__stex_notation_arity_str { #2 }
2441   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2442   \str_set:Nx \l__stex_notation_prec_str { #4 }
2443
2444   % precedences
2445   \str_if_empty:NTF \l__stex_notation_prec_str {
2446     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2447       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2448     }{
2449       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2450     }
2451   } {
2452     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2453       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2454       \int_step_inline:nn { \l__stex_notation_arity_str } {
2455         \exp_args:NNo
2456         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2457       }
2458     }{
2459       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2460       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2461         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2462         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2463           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2464             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2465           \seq_map_inline:Nn \l_tmpa_seq {
2466             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2467           }
2468         }
2469       }{
2470         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2471           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2472         }{
2473           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2474         }
2475       }
2476     }
2477   }

```

```

2478
2479 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2480 \int_step_inline:nn { \l__stex_notation_arity_str } {
2481   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2482     \exp_args:NNo
2483     \seq_put_right:No \l__stex_notation_precedences_seq {
2484       \l__stex_notation_opprec_tl
2485     }
2486   }
2487 }
2488 \tl_clear:N \l_stex_notation_dummyargs_tl
2489
2490 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2491   \exp_args:NNe
2492   \cs_set:Npn \l_stex_notation_macrocode_cs {
2493     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2494     { \l__stex_notation_suffix_str }
2495     { \l__stex_notation_opprec_tl }
2496     { \exp_not:n { #5 } }
2497   }
2498   \l_stex_notation_after_do_tl
2499 }{
2500   \str_if_in:NnTF \l__stex_notation_args_str b {
2501     \exp_args:Nne \use:nn
2502     {
2503       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2504       \cs_set:Npn \l__stex_notation_arity_str } { {
2505         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2506         { \l__stex_notation_suffix_str }
2507         { \l__stex_notation_opprec_tl }
2508         { \exp_not:n { #5 } }
2509       }}
2510   }{
2511     \str_if_in:NnTF \l__stex_notation_args_str B {
2512       \exp_args:Nne \use:nn
2513       {
2514         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2515         \cs_set:Npn \l__stex_notation_arity_str } { {
2516           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2517           { \l__stex_notation_suffix_str }
2518           { \l__stex_notation_opprec_tl }
2519           { \exp_not:n { #5 } }
2520         } }
2521     }{
2522       \exp_args:Nne \use:nn
2523       {
2524         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2525         \cs_set:Npn \l__stex_notation_arity_str } { {
2526           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2527           { \l__stex_notation_suffix_str }
2528           { \l__stex_notation_opprec_tl }
2529           { \exp_not:n { #5 } }
2530         } }
2531     }

```

```

2532     }
2533
2534     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2535     \int_zero:N \l__stex_notation_currarg_int
2536     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2537     \__stex_notation_arguments:
2538   }
2539 }

```

(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

```

2540 \cs_new_protected:Nn \__stex_notation_arguments: {
2541   \int_incr:N \l__stex_notation_currarg_int
2542   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2543     \l_stex_notation_after_do_tl
2544   }{
2545     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2546     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2547     \str_if_eq:VnTF \l_tmpa_str a {
2548       \__stex_notation_argument_assoc:nn{a}
2549     }{
2550       \str_if_eq:VnTF \l_tmpa_str B {
2551         \__stex_notation_argument_assoc:nn{B}
2552       }{
2553         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2554         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2555           { \stex_term_math_arg:nnn
2556             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2557             { \l_tmpb_str }
2558             { ###\int_use:N \l__stex_notation_currarg_int }
2559           }
2560         }
2561         \__stex_notation_arguments:
2562       }
2563     }
2564   }
2565 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2566 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2567
2568   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2569     {\l__stex_notation_arity_str}{
2570       #2
2571     }
2572   \int_zero:N \l_tmpa_int
2573   \tl_clear:N \l_tmpa_tl
2574   \str_map_inline:Nn \l__stex_notation_args_str {
2575     \int_incr:N \l_tmpa_int
2576     \tl_put_right:Nx \l_tmpa_tl {
2577       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```

```

2578     \str_if_eq:nnTF {##1}{B}{ } }{
2579         {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2580         }
2581     }
2582 }
2583 }
2584 \exp_after:wN\exp_after:wN\exp_after:wN \def
2585 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2586 \exp_after:wN\exp_after:wN\exp_after:wN ##
2587 \exp_after:wN\exp_after:wN\exp_after:wN 1
2588 \exp_after:wN\exp_after:wN\exp_after:wN ##
2589 \exp_after:wN\exp_after:wN\exp_after:wN 2
2590 \exp_after:wN\exp_after:wN\exp_after:wN {
2591     \exp_after:wN \exp_after:wN \exp_after:wN
2592     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2593         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2594     }
2595 }
2596
2597 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2598 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2599     \_stex_term_math_assoc_arg:nnnn
2600     { #1\int_use:N \l__stex_notation_currarg_int }
2601     { \l_tmpa_str }
2602     { #####\int_use:N \l__stex_notation_currarg_int }
2603     { \l_tmpa_cs {####1} {####2} }
2604 } }
2605 \__stex_notation_arguments:
2606 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2607 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2608     \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2609     \cs_set_nopar:Npn {#3}{#4}
2610     \tl_if_empty:nF {#5}{
2611         \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2612     }
2613     \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2614         \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2615     }
2616 }
2617
2618 \cs_new_protected:Nn \_stex_notation_final: {
2619
2620     \stex_execute_in_module:x {
2621         \_stex_notation_restore_notation:nnnnn
2622         {\l_stex_get_symbol_uri_str}
2623         {\l_stex_notation_suffix_str}
2624         {\l__stex_notation_arity_str}
2625     {
2626         \exp_after:wN \exp_after:wN \exp_after:wN
2627         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2628     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2629   }
2630   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2631 }
2632
2633 \stex_debug:nn{symbols}{
2634   Notation~\l__stex_notation_suffix_str
2635   ~for~\l_stex_get_symbol_uri_str^^J
2636   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2637   Argument~precedences:~
2638     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2639   Notation: \cs_meaning:c {
2640     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2641     \l__stex_notation_suffix_str
2642     _cs
2643   }
2644 }
2645 % HTML annotations
2646 \stex_if_do_html:T {
2647   \stex_annotate_invisible:nnn { notation }
2648   { \l_stex_get_symbol_uri_str } {
2649     \stex_annotate_invisible:nnn { notationfragment }
2650     { \l__stex_notation_suffix_str }{}
2651     \stex_annotate_invisible:nnn { precedence }
2652     { \l__stex_notation_prec_str }{}
2653
2654     \int_zero:N \l_tmpa_int
2655     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2656     \tl_clear:N \l_tmpa_tl
2657     \int_step_inline:nn { \l__stex_notation_arity_str }{
2658       \int_incr:N \l_tmpa_int
2659       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2660       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2661       \str_if_eq:VnTF \l_tmpb_str a {
2662         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2663           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2664           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2665         } }
2666       }{
2667         \str_if_eq:VnTF \l_tmpb_str B {
2668           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2669             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2670             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2671           } }
2672         }{
2673           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2674             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2675           } }
2676         }
2677       }
2678     }
2679     \stex_annotate_invisible:nnn { notationcomp }{}{
2680       \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2681       $ \exp_args:Nno \use:nn { \use:c {

```

```

2682         stex_notation_ \l_stex_current_symbol_str
2683         \c_hash_str \l__stex_notation_suffix_str _cs
2684     } } { \l_tmpa_tl } $
2685 }
2686 }
2687 }
2688 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2689 \keys_define:nn { stex / setnotation } {
2690   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2691   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2692   unknown .code:n      = \str_set:Nx
2693     \l__stex_notation_variant_str \l_keys_key_str
2694 }
2695
2696 \cs_new_protected:Nn \stex_setnotation_args:n {
2697   % \str_clear:N \l__stex_notation_lang_str
2698   \str_clear:N \l__stex_notation_variant_str
2699   \keys_set:nn { stex / setnotation } { #1 }
2700 }
2701
2702 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2703   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2704     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2705     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2706   }
2707 }
2708
2709 \cs_new_protected:Nn \stex_setnotation:n {
2710   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2711     { \l__stex_notation_variant_str }{
2712     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2713     \stex_debug:nn {notations}{
2714       Setting~default~notation~
2715       {\l__stex_notation_variant_str }~for~
2716       #1 \\
2717       \expandafter\meaning\csname
2718       l_stex_symdecl_#1_notations\endcsname
2719     }
2720   }{
2721     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2722   }
2723 }
2724
2725 \NewDocumentCommand \setnotation {m m} {
2726   \stex_get_symbol:n { #1 }
2727   \stex_setnotation_args:n { #2 }
2728   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2729   \stex_smsmode_do:\ignorespacesandpars
2730 }
2731

```



```

2732 \cs_new_protected:Nn \stex_copy_notations:nn {
2733   \stex_debug:nn {notations}{
2734     Copying~notations~from~#2~to~#1\\
2735     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2736   }
2737   \tl_clear:N \l_tmpa_tl
2738   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2739     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2740   }
2741   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2742     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2743     \edef \l_tmpa_tl {
2744       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2745       \exp_after:wN\exp_after:wN\exp_after:wN {
2746         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2747       }
2748     }
2749
2750     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2751     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2752     \exp_after:wN { \l_tmpa_tl }
2753
2754     \edef \l_tmpa_tl {
2755       \exp_after:wN \exp_not:n \exp_after:wN {
2756         \l_tmpa_tl {##### 1}{##### 2}
2757       }
2758     }
2759
2760     \stex_execute_in_module:x {
2761       \__stex_notation_restore_notation:nnnnn
2762       {#1}{##1}
2763       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2764       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2765       {
2766         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2767           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2768         }
2769       }
2770     }
2771   }
2772 }
2773
2774 \NewDocumentCommand \copynotation {m m} {
2775   \stex_get_symbol:n { #1 }
2776   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2777   \stex_get_symbol:n { #2 }
2778   \exp_args:Noo
2779   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2780   \stex_smsmode_do:\ignorespacesandpars
2781 }
2782

```

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

\symdef

```
2783 \keys_define:nn { stex / symdef } {
2784   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2785   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2786   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2787   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2788   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2789   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2790   op        .tl_set:N    = \l__stex_notation_op_tl ,
2791   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2792   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2793   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2794   assoc     .choices:nn =
2795     {bin,binl,binr,pre,conj,pwconj}
2796     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2797   unknown   .code:n      = \str_set:Nx
2798     \l__stex_notation_variant_str \l_keys_key_str
2799 }
2800
2801 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2802   \str_clear:N \l_stex_symdecl_name_str
2803   \str_clear:N \l_stex_symdecl_args_str
2804   \str_clear:N \l_stex_symdecl_assoctype_str
2805   \str_clear:N \l_stex_symdecl_reorder_str
2806   \bool_set_false:N \l_stex_symdecl_local_bool
2807   \tl_clear:N \l_stex_symdecl_type_tl
2808   \tl_clear:N \l_stex_symdecl_definiens_tl
2809   % \str_clear:N \l__stex_notation_lang_str
2810   \str_clear:N \l__stex_notation_variant_str
2811   \str_clear:N \l__stex_notation_prec_str
2812   \tl_clear:N \l__stex_notation_op_tl
2813
2814   \keys_set:nn { stex / symdef } { #1 }
2815 }
2816
2817 \NewDocumentCommand \symdef { m O{} } {
2818   \__stex_notation_symdef_args:n { #2 }
2819   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2820   \stex_symdecl_do:n { #1 }
2821   \tl_set:Nn \l_stex_notation_after_do_tl {
2822     \__stex_notation_final:
2823     \stex_smsmode_do:\ignorespacesandpars
2824   }
2825   \str_set:Nx \l_stex_get_symbol_uri_str {
2826     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2827   }
2828   \exp_args:Nx \stex_notation_do:nnnnn
2829     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2830     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2831     { \l__stex_notation_variant_str }
2832     { \l__stex_notation_prec_str }
2833 }
2834 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3 Variables

```

2835 <@@=stex_variables>
2836
2837 \keys_define:nn { stex / vardef } {
2838   name      .str_set_x:N = \l__stex_variables_name_str ,
2839   args      .str_set_x:N = \l__stex_variables_args_str ,
2840   type      .tl_set:N    = \l__stex_variables_type_tl ,
2841   def       .tl_set:N    = \l__stex_variables_def_tl ,
2842   op        .tl_set:N    = \l__stex_variables_op_tl ,
2843   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2844   assoc     .choices:nn =
2845     {bin,binl,binr,pre,conj,pwconj}
2846     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2847   bind      .choices:nn =
2848     {forall,exists}
2849     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2850 }
2851
2852 \cs_new_protected:Nn \__stex_variables_args:n {
2853   \str_clear:N \l__stex_variables_name_str
2854   \str_clear:N \l__stex_variables_args_str
2855   \str_clear:N \l__stex_variables_prec_str
2856   \str_clear:N \l__stex_variables_assoctype_str
2857   \str_clear:N \l__stex_variables_bind_str
2858   \tl_clear:N \l__stex_variables_type_tl
2859   \tl_clear:N \l__stex_variables_def_tl
2860   \tl_clear:N \l__stex_variables_op_tl
2861
2862   \keys_set:nn { stex / vardef } { #1 }
2863 }
2864
2865 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2866   \__stex_variables_args:n {#2}
2867   \str_if_empty:NT \l__stex_variables_name_str {
2868     \str_set:Nx \l__stex_variables_name_str { #1 }
2869   }
2870   \prop_clear:N \l_tmpa_prop
2871   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2872
2873   \int_zero:N \l_tmpb_int
2874   \bool_set_true:N \l_tmpa_bool
2875   \str_map_inline:Nn \l__stex_variables_args_str {
2876     \token_case_meaning:NnF ##1 {
2877       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2878       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2879       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2880       {\tl_to_str:n a} {
2881         \bool_set_false:N \l_tmpa_bool
2882         \int_incr:N \l_tmpb_int
2883       }

```

```

2884     {\tl_to_str:n B} {
2885       \bool_set_false:N \l_tmpa_bool
2886       \int_incr:N \l_tmpb_int
2887     }
2888   }{
2889     \msg_error:nxxx{stex}{error/wrongargs}{
2890       variable~\l__stex_variables_name_str
2891     }{##1}
2892   }
2893 }
2894 \bool_if:NTF \l_tmpa_bool {
2895   % possibly numeric
2896   \str_if_empty:NTF \l__stex_variables_args_str {
2897     \prop_put:Nnn \l_tmpa_prop { args } {}
2898     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2899   }{
2900     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2901     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2902     \str_clear:N \l_tmpa_str
2903     \int_step_inline:nn \l_tmpa_int {
2904       \str_put_right:Nn \l_tmpa_str i
2905     }
2906     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2907     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2908   }
2909 } {
2910   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2911   \prop_put:Nnx \l_tmpa_prop { arity }
2912     { \str_count:N \l__stex_variables_args_str }
2913 }
2914 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2915 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2916
2917 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2918
2919 \tl_if_empty:NF \l__stex_variables_op_tl {
2920   \cs_set:cpx {
2921     stex_var_op_notation_\l__stex_variables_name_str _cs
2922   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2923 }
2924
2925 \tl_set:Nn \l_stex_notation_after_do_tl {
2926   \exp_args:Nne \use:nn {
2927     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2928     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2929   } {{
2930     \exp_after:wN \exp_after:wN \exp_after:wN
2931     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2932     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2933   }}
2934 \stex_if_do_html:T {
2935   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2936     \stex_annotate_invisible:nnn { precedence }
2937     { \l__stex_variables_prec_str }{}

```

```

2938 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l
2939 \stex_annotate_invisible:nnn{args}{}}{\l__stex_variables_args_str }
2940 \stex_annotate_invisible:nnn{macroname}{#1}{ }
2941 \tl_if_empty:NF \l__stex_variables_def_tl {
2942   \stex_annotate_invisible:nnn{definiens}{ }
2943   {\$ \l__stex_variables_def_tl$}
2944 }
2945 \str_if_empty:NF \l__stex_variables_assoctype_str {
2946   \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{ }
2947 }
2948 \str_if_empty:NF \l__stex_variables_bind_str {
2949   \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{ }
2950 }
2951 \int_zero:N \l_tmpa_int
2952 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2953 \tl_clear:N \l_tmpa_tl
2954 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2955   \int_incr:N \l_tmpa_int
2956   \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2957   \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2958   \str_if_eq:VnTF \l_tmpb_str a {
2959     \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2960       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2961       \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2962     } }
2963   }{
2964     \str_if_eq:VnTF \l_tmpb_str B {
2965       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2966         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2967         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2968       } }
2969     }{
2970       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2971         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
2972       } }
2973     }
2974   }
2975 }
2976 \stex_annotate_invisible:nnn { notationcomp }{ }{
2977   \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2978   $ \exp_args:Nno \use:nn { \use:c {
2979     stex_var_notation_\l__stex_variables_name_str_cs
2980   } } { \l_tmpa_tl } $
2981 }
2982 }
2983 }\ignorespacesandpars
2984 }
2985
2986 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2987 }
2988
2989 \cs_new:Nn \_stex_reset:N {
2990   \tl_if_exist:NTF #1 {
2991     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```

```

2992 }{
2993   \let \exp_not:N #1 \exp_not:N \undefined
2994 }
2995 }
2996
2997 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2998   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2999   \exp_args:Nnx \use:nn {
3000     % TODO
3001     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3002       #2
3003     }
3004   }{
3005     \stex_reset:N \varnot
3006     \stex_reset:N \vartype
3007     \stex_reset:N \vardefi
3008   }
3009 }
3010
3011 \NewDocumentCommand \vardef { s } {
3012   \IfBooleanTF#1 {
3013     \__stex_variables_do_complex:nn
3014   }{
3015     \__stex_variables_do_simple:nnn
3016   }
3017 }
3018
3019 \NewDocumentCommand \svar { 0{} m }{
3020   \tl_if_empty:nTF {#1}{
3021     \str_set:Nn \l_tmpa_str { #2 }
3022   }{
3023     \str_set:Nn \l_tmpa_str { #1 }
3024   }
3025   \stex_term_omv:nn {
3026     var://\l_tmpa_str
3027   }{
3028     \exp_args:Nnx \use:nn {
3029       \def\comp{\_varcomp}
3030       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3031       \comp{ #2 }
3032     }{
3033       \stex_reset:N \comp
3034       \stex_reset:N \l_stex_current_symbol_str
3035     }
3036   }
3037 }
3038
3039
3040
3041 \keys_define:nn { stex / varseq } {
3042   name .str_set_x:N = \l__stex_variables_name_str ,
3043   args .int_set:N   = \l__stex_variables_args_int ,
3044   type .tl_set:N    = \l__stex_variables_type_tl ,
3045   mid .tl_set:N     = \l__stex_variables_mid_tl ,

```

```

3046   bind      .choices:nn      =
3047       {forall,exists}
3048       {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3049   }
3050
3051   \cs_new_protected:Nn \__stex_variables_seq_args:n {
3052     \str_clear:N \l__stex_variables_name_str
3053     \int_set:Nn \l__stex_variables_args_int 1
3054     \tl_clear:N \l__stex_variables_type_tl
3055     \str_clear:N \l__stex_variables_bind_str
3056
3057     \keys_set:nn { stex / varseq } { #1 }
3058   }
3059
3060   \NewDocumentCommand \varseq {m O{}} m m m m){
3061     \__stex_variables_seq_args:n { #2 }
3062     \str_if_empty:NT \l__stex_variables_name_str {
3063       \str_set:Nx \l__stex_variables_name_str { #1 }
3064     }
3065     \prop_clear:N \l_tmpa_prop
3066     \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3067
3068     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3069     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3070       \msg_error:nnxx{stex}{error/seqlength}
3071       {\int_use:N \l__stex_variables_args_int}
3072       {\seq_count:N \l_tmpa_seq}
3073     }
3074     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3075     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3076       \msg_error:nnxx{stex}{error/seqlength}
3077       {\int_use:N \l__stex_variables_args_int}
3078       {\seq_count:N \l_tmpb_seq}
3079     }
3080     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3081     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3082
3083     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3084     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3085
3086     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3087     \int_step_inline:nn \l__stex_variables_args_int {
3088       \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3089     }
3090     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3091     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3092     \tl_if_empty:NF \l__stex_variables_mid_tl {
3093       \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3094       \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3095     }
3096     \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3097     \int_step_inline:nn \l__stex_variables_args_int {
3098       \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3099     }

```

```

3100 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3101 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3102
3103
3104 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3105
3106 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3107
3108 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3109
3110 \int_step_inline:nn \l__stex_variables_args_int {
3111   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3112     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3113   }}
3114 }
3115
3116 \tl_set:Nx \l_tmpa_tl {
3117   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3118     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3119   }
3120 }
3121
3122 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3123
3124 \exp_args:Nno \use:nn {
3125   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3126   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3127
3128   \stex_debug:nn{sequences}{New~Sequence:~
3129     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3130     \prop_to_keyval:N \l_tmpa_prop
3131   }
3132   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3133     \tl_if_empty:NF \l__stex_variables_type_tl {
3134       \stex_annotate:nnn {type}{}{\${\seqtype\l__stex_variables_type_tl$}
3135     }
3136     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3137     \str_if_empty:NF \l__stex_variables_bind_str {
3138       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3139     }
3140   }}
3141
3142   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3143   \ignorespacesandpars
3144 }
3145
3146 </package>

```


Chapter 30

STEX -Terms Implementation

```
3147 <*package>
3148
3149 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3150
3151 <@@=stex_terms>
3152
3153 Warnings and error messages
3154 \msg_new:nnn{stex}{error/nonotation}{
3155   Symbol~#1~invoked,~but~has~no~notation#2!
3156 }
3157 \msg_new:nnn{stex}{error/notationarg}{
3158   Error~in~parsing~notation~#1
3159 }
3160 \msg_new:nnn{stex}{error/noop}{
3161   Symbol~#1~has~no~operator~notation~for~notation~#2
3162 }
3163 \msg_new:nnn{stex}{error/notallowed}{
3164   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3165 }
3166 \msg_new:nnn{stex}{error/doubleargument}{
3167   Argument~#1~of~symbol~#2~already~assigned
3168 }
3169 \msg_new:nnn{stex}{error/overarity}{
3170   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3171 }
3172
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3171
3172
3173 \bool_new:N \l_stex_allow_semantic_bool
3174 \bool_set_true:N \l_stex_allow_semantic_bool
3175
```

```

3176 \cs_new_protected:Nn \stex_invoke_symbol:n {
3177   \bool_if:NTF \l_stex_allow_semantic_bool {
3178     \str_if_eq:eeF {
3179       \prop_item:cn {
3180         l_stex_symdecl_#1_prop
3181       }{ deprecate }
3182     }{}{
3183       \msg_warning:nxxx{stex}{warning/deprecated}{
3184         Symbol~#1
3185       }{
3186         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3187       }
3188     }
3189     \if_mode_math:
3190       \exp_after:wN \__stex_terms_invoke_math:n
3191     \else:
3192       \exp_after:wN \__stex_terms_invoke_text:n
3193     \fi: { #1 }
3194   }{
3195     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3196   }
3197 }
3198
3199 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3200   \peek_charcode_remove:NTF ! {
3201     \__stex_terms_invoke_op_custom:nn {#1}
3202   }{
3203     \__stex_terms_invoke_custom:nn {#1}
3204   }
3205 }
3206
3207 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3208   \peek_charcode_remove:NTF ! {
3209     % operator
3210     \peek_charcode_remove:NTF * {
3211       % custom op
3212       \__stex_terms_invoke_op_custom:nn {#1}
3213     }{
3214       % op notation
3215       \peek_charcode:NTF [ {
3216         \__stex_terms_invoke_op_notation:nw {#1}
3217       }{
3218         \__stex_terms_invoke_op_notation:nw {#1}[]
3219       }
3220     }
3221   }{
3222     \peek_charcode_remove:NTF * {
3223       \__stex_terms_invoke_custom:nn {#1}
3224       % custom
3225     }{
3226       % normal
3227       \peek_charcode:NTF [ {
3228         \__stex_terms_invoke_notation:nw {#1}
3229       }{

```

```

3230     \__stex_terms_invoke_notation:nw {#1}[]
3231   }
3232 }
3233 }
3234 }
3235
3236
3237 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3238   \exp_args:Nnx \use:nn {
3239     \def\comp{\_comp}
3240     \str_set:Nn \l_stex_current_symbol_str { #1 }
3241     \bool_set_false:N \l_stex_allow_semantic_bool
3242     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3243       \comp{ #2 }
3244     }
3245   }{
3246     \stex_reset:N \comp
3247     \stex_reset:N \l_stex_current_symbol_str
3248     \bool_set_true:N \l_stex_allow_semantic_bool
3249   }
3250 }
3251
3252 \keys_define:nn { stex / terms } {
3253   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3254   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3255   unknown .code:n = \str_set:Nx
3256     \l_stex_notation_variant_str \l_keys_key_str
3257 }
3258
3259 \cs_new_protected:Nn \__stex_terms_args:n {
3260   % \str_clear:N \l_stex_notation_lang_str
3261   \str_clear:N \l_stex_notation_variant_str
3262
3263   \keys_set:nn { stex / terms } { #1 }
3264 }
3265
3266 \cs_new_protected:Nn \stex_find_notation:nn {
3267   \__stex_terms_args:n { #2 }
3268   \seq_if_empty:cTF {
3269     l_stex_symdecl_ #1 _notations
3270   } {
3271     \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3272   } {
3273     \str_if_empty:NTF \l_stex_notation_variant_str {
3274       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3275     }{
3276       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3277         \l_stex_notation_variant_str
3278       }{
3279         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3280       }{
3281         \msg_error:nxxx{stex}{error/nonotation}{#1}{~\l_stex_notation_variant_str
3282       }
3283     }

```

```

3284     }
3285   }
3286 }
3287 }
3288
3289 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3290   \exp_args:Nnx \use:nn {
3291     \def\comp{\_comp}
3292     \str_set:Nn \l_stex_current_symbol_str { #1 }
3293     \stex_find_notation:nn { #1 }{ #2 }
3294     \bool_set_false:N \l_stex_allow_semantic_bool
3295     \cs_if_exist:cTF {
3296       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3297     }{
3298       \_stex_term_oms:nnn { #1 }{
3299         #1 \c_hash_str \l_stex_notation_variant_str
3300       }{
3301         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3302       }
3303     }{
3304       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3305         \cs_if_exist:cTF {
3306           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3307         }{
3308           \tl_set:Nx \stex_symbol_after_invokation_tl {
3309             \_stex_reset:N \comp
3310             \_stex_reset:N \stex_symbol_after_invokation_tl
3311             \_stex_reset:N \l_stex_current_symbol_str
3312             \bool_set_true:N \l_stex_allow_semantic_bool
3313           }
3314           \def\comp{\_comp}
3315           \str_set:Nn \l_stex_current_symbol_str { #1 }
3316           \bool_set_false:N \l_stex_allow_semantic_bool
3317           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3318         }{
3319           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3320             ~\l_stex_notation_variant_str
3321           }
3322         }
3323       }{
3324         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3325       }
3326     }
3327   }{
3328     \_stex_reset:N \comp
3329     \_stex_reset:N \l_stex_current_symbol_str
3330     \bool_set_true:N \l_stex_allow_semantic_bool
3331   }
3332 }
3333
3334 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3335   \stex_find_notation:nn { #1 }{ #2 }
3336   \cs_if_exist:cTF {
3337     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3338 }{
3339   \tl_set:Nx \stex_symbol_after_invokation_tl {
3340     \_stex_reset:N \comp
3341     \_stex_reset:N \stex_symbol_after_invokation_tl
3342     \_stex_reset:N \l_stex_current_symbol_str
3343     \bool_set_true:N \l_stex_allow_semantic_bool
3344   }
3345   \def\comp{\_comp}
3346   \str_set:Nn \l_stex_current_symbol_str { #1 }
3347   \bool_set_false:N \l_stex_allow_semantic_bool
3348   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3349 }{
3350   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3351     ~\l_stex_notation_variant_str
3352   }
3353 }
3354 }
3355
3356 \prop_new:N \l__stex_terms_custom_args_prop
3357
3358 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3359   \exp_args:Nnx \use:nn {
3360     \bool_set_false:N \l_stex_allow_semantic_bool
3361     \def\comp{\_comp}
3362     \str_set:Nn \l_stex_current_symbol_str { #1 }
3363     \prop_clear:N \l__stex_terms_custom_args_prop
3364     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3365     \prop_get:cnN {
3366       l_stex_symdecl_#1 _prop
3367     }{ args } \l_tmpa_str
3368     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3369     \tl_set:Nn \arg { \__stex_terms_arg: }
3370     \str_if_empty:NTF \l_tmpa_str {
3371       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3372     }{
3373       \str_if_in:NnTF \l_tmpa_str b {
3374         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3375       }{
3376         \str_if_in:NnTF \l_tmpa_str B {
3377           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3378         }{
3379           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3380         }
3381       }
3382     }
3383     % TODO check that all arguments exist
3384   }{
3385     \_stex_reset:N \l_stex_current_symbol_str
3386     \_stex_reset:N \arg
3387     \_stex_reset:N \comp
3388     \_stex_reset:N \l__stex_terms_custom_args_prop
3389     \bool_set_true:N \l_stex_allow_semantic_bool
3390   }
3391 }

```

```

3392
3393 \NewDocumentCommand \__stex_terms_arg: { s 0{} m}{
3394   \tl_if_empty:nTF {#2}{
3395     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3396     \bool_set_true:N \l_tmpa_bool
3397     \bool_do_while:Nn \l_tmpa_bool {
3398       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3399         \int_incr:N \l_tmpa_int
3400       }{
3401         \bool_set_false:N \l_tmpa_bool
3402       }
3403     }
3404   }{
3405     \int_set:Nn \l_tmpa_int { #2 }
3406   }
3407   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3408   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3409     \msg_error:nnxxx{stex}{error/overarity}
3410     {\int_use:N \l_tmpa_int}
3411     {\l_stex_current_symbol_str}
3412     {\str_count:N \l_tmpa_str}
3413   }
3414   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3415   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3416     \bool_lazy_any:nF {
3417       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3418       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3419     }{
3420       \msg_error:nnxx{stex}{error/doubleargument}
3421       {\int_use:N \l_tmpa_int}
3422       {\l_stex_current_symbol_str}
3423     }
3424   }
3425   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3426   \bool_set_true:N \l_stex_allow_semantic_bool
3427   \IfBooleanTF#1{
3428     \stex_annotate_invisible:n { %TODO
3429       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3430     }
3431   }{ %TODO
3432     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3433   }
3434   \bool_set_false:N \l_stex_allow_semantic_bool
3435 }
3436
3437
3438 \cs_new_protected:Nn \_stex_term_arg:nn {
3439   \bool_set_true:N \l_stex_allow_semantic_bool
3440   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3441   \bool_set_false:N \l_stex_allow_semantic_bool
3442 }
3443
3444 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3445   \exp_args:Nnx \use:nn

```

```

3446 { \int_set:Nn \l__stex_terms_downprec { #2 }
3447   \stex_term_arg:nn { #1 }{ #3 }
3448 }
3449 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3450 }

```

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

\stex_term_math_assoc_arg:nnnn

```

3451 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3452   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3453   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3454   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3455     \expandafter\if\expandafter\relax\noexpand#3
3456     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3457   }else
3458     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3459   \fi
3460   \l_tmpa_tl
3461 }{
3462   \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3463 }
3464 }
3465
3466 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3467   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3468   \str_if_empty:NTF \l_tmpa_str {
3469     \exp_args:Nx \cs_if_eq:NNTF {
3470       \tl_head:N #1
3471     } \stex_invoke_sequence:n {
3472       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3473       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3474       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3475       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3476       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3477         \exp_not:n{\exp_args:Nnx \use:nn} {
3478           \exp_not:n {
3479             \def\comp{\_varcomp}
3480             \str_set:Nn \l_stex_current_symbol_str
3481             } {varseq://\l_tmpa_str}
3482             \exp_not:n{ ##1 }
3483           }{
3484             \exp_not:n {
3485               \stex_reset:N \comp
3486               \stex_reset:N \l_stex_current_symbol_str
3487             }
3488           }
3489         }}}
3490       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3491       \seq_reverse:N \l_tmpa_seq
3492       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3493       \seq_map_inline:Nn \l_tmpa_seq {
3494         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3495           \exp_args:Nno

```

```

3496         \l_tmpa_cs { ##1 } \l_tmpa_tl
3497     }
3498 }
3499 \tl_set:Nx \l_tmpa_tl {
3500     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3501         \exp_args:No \exp_not:n \l_tmpa_tl
3502     }
3503 }
3504 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3505 }{
3506     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3507 }
3508 } {
3509     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3510 }
3511 }
3512 }
3513
3514 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3515     \clist_set:Nn \l_tmpa_clist{ #2 }
3516     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3517         \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3518     }{
3519         \clist_reverse:N \l_tmpa_clist
3520         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3521         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3522             \exp_args:No \exp_not:n \l_tmpa_tl
3523         }}
3524         \clist_map_inline:Nn \l_tmpa_clist {
3525             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3526                 \exp_args:Nno
3527                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3528             }
3529         }
3530     }
3531     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3532 }

```

(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
3533 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3534 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3535 \int_new:N \l__stex_terms_downprec
3536 \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

```
3537 \tl_set:Nn \l__stex_terms_left_bracket_str (
3538 \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

```
3539 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3540   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3541     \bool_set_false:N \l__stex_terms_brackets_done_bool
3542     #2
3543   } {
3544     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3545       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3546         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3547         \dobrackets { #2 }
3548       }
3549     }{ #2 }
3550   }
3551 }
```

(End definition for __stex_terms_maybe_brackets:nn.)

\dobrackets

```
3552 \bool_new:N \l__stex_terms_brackets_done_bool
3553 %\RequirePackage{scalerel}
3554 \cs_new_protected:Npn \dobrackets #1 {
3555   %\ThisStyle{\if D\m@switch
3556   %   \exp_args:Nnx \use:nn
3557   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3558   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3559   % \else
3560   \exp_args:Nnx \use:nn
3561   {
3562     \bool_set_true:N \l__stex_terms_brackets_done_bool
3563     \int_set:Nn \l__stex_terms_downprec \infprec
3564     \l__stex_terms_left_bracket_str
3565     #1
3566   }
3567   {
3568     \bool_set_false:N \l__stex_terms_brackets_done_bool
3569     \l__stex_terms_right_bracket_str
3570     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3571   }
3572   %\fi}
3573 }
```

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

\withbrackets

```
3574 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3575   \exp_args:Nnx \use:nn
3576   {
3577     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```

3578 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3579 #3
3580 }
3581 {
3582 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3583 {\l__stex_terms_left_bracket_str}
3584 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3585 {\l__stex_terms_right_bracket_str}
3586 }
3587 }

```

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

`\STEXinvisible`

```

3588 \cs_new_protected:Npn \STEXinvisible #1 {
3589 \stex_annotate_invisible:n { #1 }
3590 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3591 \cs_new_protected:Nn \_stex_term_oms:nnn {
3592 \stex_annotate:nnn{ OMID }{ #2 }{
3593 #3
3594 }
3595 }
3596
3597 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3598 \_stex_terms_maybe_brackets:nn { #3 }{
3599 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3600 }
3601 }

```

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

`_stex_term_math_omv:nn`

```

3602 \cs_new_protected:Nn \_stex_term_omv:nn {
3603 \stex_annotate:nnn{ OMV }{ #1 }{
3604 #2
3605 }
3606 }

```

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

`_stex_term_math_oma:nnnn`

```

3607 \cs_new_protected:Nn \_stex_term_oma:nnn {
3608 \stex_annotate:nnn{ OMA }{ #2 }{
3609 #3
3610 }
3611 }
3612
3613 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3614 \_stex_terms_maybe_brackets:nn { #3 }{
3615 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

3616 }
3617 }

```

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

`_stex_term_math_omb:nnnn`

```

3618 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3619   \stex_annotate:nnn{ OMBIND }{ #2 }{
3620     #3
3621   }
3622 }
3623
3624 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3625   \__stex_terms_maybe_brackets:nn { #3 }{
3626     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3627   }
3628 }

```

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

`\symref`
`\symname`

```

3629 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3630
3631 \keys_define:nn { stex / symname } {
3632   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3633   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3634   root     .tl_set_x:N = \l__stex_terms_root_tl
3635 }
3636
3637 \cs_new_protected:Nn \stex_symname_args:n {
3638   \tl_clear:N \l__stex_terms_post_tl
3639   \tl_clear:N \l__stex_terms_pre_tl
3640   \tl_clear:N \l__stex_terms_root_str
3641   \keys_set:nn { stex / symname } { #1 }
3642 }
3643
3644 \NewDocumentCommand \symref { m m }{
3645   \let\compemph_uri_prev:\compemph@uri
3646   \let\compemph@uri\symrefemph@uri
3647   \STEXsymbol{#1}!{ #2 }
3648   \let\compemph@uri\compemph_uri_prev:
3649 }
3650
3651 \NewDocumentCommand \synonym { 0{} m m }{
3652   \stex_symname_args:n { #1 }
3653   \let\compemph_uri_prev:\compemph@uri
3654   \let\compemph@uri\symrefemph@uri
3655   % TODO
3656   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3657   \let\compemph@uri\compemph_uri_prev:
3658 }
3659
3660 \NewDocumentCommand \symname { 0{} m }{
3661   \stex_symname_args:n { #1 }
3662   \stex_get_symbol:n { #2 }

```

```

3663 \str_set:Nx \l_tmpa_str {
3664   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3665 }
3666 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3667
3668 \let\compemph_uri_prev:\compemph@uri
3669 \let\compemph@uri\symrefemph@uri
3670 \exp_args:NNx \use:nn
3671 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3672   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3673 } }
3674 \let\compemph@uri\compemph_uri_prev:
3675 }
3676
3677 \NewDocumentCommand \Symname { 0{} m }{
3678   \stex_symname_args:n { #1 }
3679   \stex_get_symbol:n { #2 }
3680   \str_set:Nx \l_tmpa_str {
3681     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3682   }
3683   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3684   \let\compemph_uri_prev:\compemph@uri
3685   \let\compemph@uri\symrefemph@uri
3686   \exp_args:NNx \use:nn
3687   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3688     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3689     \l__stex_terms_post_tl
3690   } }
3691   \let\compemph@uri\compemph_uri_prev:
3692 }

```

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

30.3 Notation Components

```

3693 <@@=stex_notationcomps>
3694
3695 \comp
3696 \compemph@uri
3697 \compemph
3698 \defemph
3699 \defemph@uri
3700 \symrefemph
3701 \symrefemph@uri
3702 \varemp
3703 \varemp@uri
3704
3705 \cs_new_protected:Npn \_comp #1 {
3706   \str_if_empty:NF \l_stex_current_symbol_str {
3707     \stex_html_backend:TF {
3708       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3709     }{
3710       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3711     }
3712   }
3713 }
3714
3715 \cs_new_protected:Npn \_varcomp #1 {
3716   \str_if_empty:NF \l_stex_current_symbol_str {
3717     \stex_html_backend:TF {
3718       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3719     }{
3720       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3721     }
3722   }
3723 }

```

```

3710     }
3711   }
3712 }
3713
3714 \def\comp{\_comp}
3715
3716 \cs_new_protected:Npn \compemph@uri #1 #2 {
3717   \compemph{ #1 }
3718 }
3719
3720
3721 \cs_new_protected:Npn \compemph #1 {
3722   #1
3723 }
3724
3725 \cs_new_protected:Npn \defemph@uri #1 #2 {
3726   \defemph{#1}
3727 }
3728
3729 \cs_new_protected:Npn \defemph #1 {
3730   \textbf{#1}
3731 }
3732
3733 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3734   \symrefemph{#1}
3735 }
3736
3737 \cs_new_protected:Npn \symrefemph #1 {
3738   \emph{#1}
3739 }
3740
3741 \cs_new_protected:Npn \varemp@uri #1 #2 {
3742   \varemp{#1}
3743 }
3744
3745 \cs_new_protected:Npn \varemp #1 {
3746   #1
3747 }

```

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

\ellipses

```

3748 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3749 \bool_new:N \l_stex_inparray_bool
3750 \bool_set_false:N \l_stex_inparray_bool
3751 \NewDocumentCommand \parray { m m } {
3752   \begingroup
3753   \bool_set_true:N \l_stex_inparray_bool
3754   \begin{array}{#1}
3755     #2
3756   \end{array}

```

```

3757 \endgroup
3758 }
3759
3760 \NewDocumentCommand \prmatrix { m } {
3761   \begingroup
3762   \bool_set_true:N \l_stex_inarray_bool
3763   \begin{matrix}
3764     #1
3765   \end{matrix}
3766   \endgroup
3767 }
3768
3769 \def \maybepline {
3770   \bool_if:NT \l_stex_inarray_bool {\hline}
3771 }
3772
3773 \def \parrayline #1 #2 {
3774   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3775 }
3776
3777 \def \pmrow #1 { \parrayline{}{ #1 } }
3778
3779 \def \parraylineh #1 #2 {
3780   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3781 }
3782
3783 \def \parraycell #1 {
3784   #1 \bool_if:NT \l_stex_inarray_bool {&}
3785 }

```

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

30.4 Variables

```

3786 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3787 \cs_new_protected:Nn \stex_invoke_variable:n {
3788   \if_mode_math:
3789     \exp_after:wN \__stex_variables_invoke_math:n
3790   \else:
3791     \exp_after:wN \__stex_variables_invoke_text:n
3792   \fi: {#1}
3793 }
3794
3795 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3796   %TODO
3797 }
3798
3799
3800 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3801   \peek_charcode_remove:NTF ! {
3802     \peek_charcode_remove:NTF ! {
3803       \peek_charcode:NTF [ {

```

```

3804     \__stex_variables_invoke_op_custom:nw
3805   }{
3806     % TODO throw error
3807   }
3808 }{
3809   \__stex_variables_invoke_op:n { #1 }
3810 }
3811 }{
3812   \peek_charcode_remove:NTF * {
3813     \__stex_variables_invoke_text:n { #1 }
3814   }{
3815     \__stex_variables_invoke_math_ii:n { #1 }
3816   }
3817 }
3818 }
3819
3820 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3821   \cs_if_exist:cTF {
3822     stex_var_op_notation_ #1 _cs
3823   }{
3824     \exp_args:Nnx \use:nn {
3825       \def\comp{\_varcomp}
3826       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3827       \_stex_term_omv:nn { var://#1 }{
3828         \use:c{stex_var_op_notation_ #1 _cs }
3829       }
3830     }{
3831       \_stex_reset:N \comp
3832       \_stex_reset:N \l_stex_current_symbol_str
3833     }
3834   }{
3835     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3836       \__stex_variables_invoke_math_ii:n {#1}
3837     }{
3838       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3839     }
3840   }
3841 }
3842
3843 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3844   \cs_if_exist:cTF {
3845     stex_var_notation_#1_cs
3846   }{
3847     \tl_set:Nx \stex_symbol_after_invokation_tl {
3848       \_stex_reset:N \comp
3849       \_stex_reset:N \stex_symbol_after_invokation_tl
3850       \_stex_reset:N \l_stex_current_symbol_str
3851       \bool_set_true:N \l_stex_allow_semantic_bool
3852     }
3853     \def\comp{\_varcomp}
3854     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3855     \bool_set_false:N \l_stex_allow_semantic_bool
3856     \use:c{stex_var_notation_#1_cs}
3857   }{

```

```

3858 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3859 }
3860 }

```

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

30.5 Sequences

```

3861 <@@=stex_sequences>
3862
3863 \cs_new_protected:Nn \stex_invoke_sequence:n {
3864   \peek_charcode_remove:NTF ! {
3865     \stex_term_omv:nn {varseq://#1}{
3866       \exp_args:Nnx \use:nn {
3867         \def\comp{\_varcomp}
3868         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3869         \prop_item:cn{stex_varseq_#1_prop}{notation}
3870       }{
3871         \stex_reset:N \comp
3872         \stex_reset:N \l_stex_current_symbol_str
3873       }
3874     }
3875   }{
3876     \bool_set_false:N \l_stex_allow_semantic_bool
3877     \def\comp{\_varcomp}
3878     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3879     \tl_set:Nx \stex_symbol_after_invokation_tl {
3880       \stex_reset:N \comp
3881       \stex_reset:N \stex_symbol_after_invokation_tl
3882       \stex_reset:N \l_stex_current_symbol_str
3883       \bool_set_true:N \l_stex_allow_semantic_bool
3884     }
3885     \use:c { stex_varseq_#1_cs }
3886   }
3887 }
3888 </package>

```


Chapter 31

STEX -Structural Features Implementation

```
3889 ⟨*package⟩
3890
3891 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3892
3893     Warnings and error messages
3894 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3895     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3896 }
3897 \msg_new:nnn{stex}{error/interpretmodule/nodefiniens}{
3898     Symbol~#1~not~assigned~in~interpretmodule~#2
3899 }
3900 \msg_new:nnn{stex}{error/unknownstructure}{
3901     No~structure~#1~found!
3902 }
3903
3904 \msg_new:nnn{stex}{error/unknownfield}{
3905     No~field~#1~in~instance~#2~found!~\#3
3906 }
3907
3908 \msg_new:nnn{stex}{error/keyval}{
3909     Invalid~key=value~pair~#1
3910 }
3911 \msg_new:nnn{stex}{error/instantiate/missing}{
3912     Assignments~missing~in~instantiate:~#1
3913 }
3914 \msg_new:nnn{stex}{error/incompatible}{
3915     Incompatible~signature:~#1~(#2)~and~#3~(#4)
3916 }
3917
```

31.1 Imports with modification

```

3918 <@@=stex_copymodule>
3919 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3920   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3921     \tl_set:Nn \l_tmpa_tl { #1 }
3922     \__stex_copymodule_get_symbol_from_cs:
3923   }{
3924     % argument is a string
3925     % is it a command name?
3926     \cs_if_exist:cTF { #1 }{
3927       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3928       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3929       \str_if_empty:NNTF \l_tmpa_str {
3930         \exp_args:Nx \cs_if_eq:NNTF {
3931           \tl_head:N \l_tmpa_tl
3932         } \stex_invoke_symbol:n {
3933           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3934         }{
3935           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3936         }
3937       } {
3938         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3939       }
3940     }{
3941       % argument is not a command name
3942       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3943       % \l_stex_all_symbols_seq
3944     }
3945   }
3946 }
3947
3948 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3949   \str_set:Nn \l_tmpa_str { #1 }
3950   \bool_set_false:N \l_tmpa_bool
3951   \bool_if:NF \l_tmpa_bool {
3952     \tl_set:Nn \l_tmpa_tl {
3953       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3954     }
3955     \str_set:Nn \l_tmpa_str { #1 }
3956     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3957     \seq_map_inline:Nn #2 {
3958       \str_set:Nn \l_tmpb_str { ##1 }
3959       \str_if_eq:eeT { \l_tmpa_str } {
3960         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3961       } {
3962         \seq_map_break:n {
3963           \tl_set:Nn \l_tmpa_tl {
3964             \str_set:Nn \l_stex_get_symbol_uri_str {
3965               ##1
3966             }
3967           }
3968         }
3969       }

```

```

3970     }
3971     \l_tmpa_tl
3972   }
3973 }
3974
3975 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3976   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3977     { \tl_tail:N \l_tmpa_tl }
3978   \tl_if_single:NTF \l_tmpa_tl {
3979     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3980       \exp_after:wN \str_set:Nn \exp_after:wN
3981         \l_stex_get_symbol_uri_str \l_tmpa_tl
3982       \__stex_copymodule_get_symbol_check:n { #1 }
3983     }{
3984       % TODO
3985       % tail is not a single group
3986     }
3987   }{
3988     % TODO
3989     % tail is not a single group
3990   }
3991 }
3992
3993 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3994   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3995     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3996       :~\seq_use:Nn #1 {,~}
3997     }
3998   }
3999 }
4000
4001 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4002   % import module
4003   \stex_import_module_uri:nn { #1 } { #2 }
4004   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4005   \stex_import_require_module:nnnn
4006     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4007     { \l_stex_import_path_str } { \l_stex_import_name_str }
4008
4009   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4010   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4011
4012   % fields
4013   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4014   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4015     \seq_map_inline:cn {c_stex_module_##1_constants}{
4016       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4017         ##1 ? ####1
4018       }
4019     }
4020   }
4021
4022   % setup prop
4023   \seq_clear:N \l_tmpa_seq

```

```

4024 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4025   name      = \l_stex_current_copymodule_name_str ,
4026   module    = \l_stex_current_module_str ,
4027   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4028   includes  = \l_tmpa_seq %,
4029 % fields    = \l_tmpa_seq
4030 }
4031 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4032   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4033 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {
4034 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {
4035
4036 \stex_if_do_html:T {
4037   \begin{stex_annotate_env} {#4} {
4038     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4039   }
4040   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{
4041 }
4042 }
4043
4044 \cs_new_protected:Nn \stex_copymodule_end:n {
4045   % apply to every field
4046   \def \l_tmpa_cs ##1 ##2 {#1}
4047
4048   \tl_clear:N \__stex_copymodule_module_tl
4049   \tl_clear:N \__stex_copymodule_exec_tl
4050
4051   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4052   \seq_clear:N \__stex_copymodule_fields_seq
4053
4054   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4055     \seq_map_inline:cn {c_stex_module_##1_constants}{
4056
4057       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4058       \l_tmpa_cs{##1}{####1}
4059
4060       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4061         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?#
4062         \stex_if_do_html:T {
4063           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4064             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?###
4065           }
4066         }
4067       }{
4068         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4069       }
4070
4071       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4072       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4073       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4074
4075       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4076         \stex_if_do_html:T {
4077           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4078         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4079     }
4080 }
4081 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4082 }
4083
4084 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4085 \tl_put_right:Nx \__stex_copymodule_module_tl {
4086     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4087     \prop_set_from_keyval:cn {
4088         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4089     }{
4090         \prop_to_keyval:N \l_tmpa_prop
4091     }
4092 }
4093
4094 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4095     \stex_if_do_html:T {
4096         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4097             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4098         }
4099     }
4100     \tl_put_right:Nx \__stex_copymodule_module_tl {
4101         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4102             \stex_invoke_symbol:n {
4103                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4104             }
4105         }
4106     }
4107 }
4108
4109 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4110
4111 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4112     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4113 }
4114
4115 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4116     \stex_if_do_html:TF{
4117         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4118     }{
4119         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4120     }
4121 }
4122 }
4123 }
4124
4125
4126 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4127 \tl_put_left:Nx \__stex_copymodule_module_tl {
4128     \prop_set_from_keyval:cn {
4129         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4130     }{
4131         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4132     }
4133 }
4134
4135 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4136   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4137 }
4138
4139 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4140 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4141 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4142
4143 \__stex_copymodule_exec_tl
4144 \stex_if_do_html:T {
4145   \end{stex_annotate_env}
4146 }
4147 }
4148
4149 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4150   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4151   \stex_deactivate_macro:Nn \symdecl {module~environments}
4152   \stex_deactivate_macro:Nn \symdef {module~environments}
4153   \stex_deactivate_macro:Nn \notation {module~environments}
4154   \stex_reactivate_macro:N \assign
4155   \stex_reactivate_macro:N \renamedekl
4156   \stex_reactivate_macro:N \donotcopy
4157   \stex_smsmode_do:
4158 }{
4159   \stex_copymodule_end:n {}
4160 }
4161
4162 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4163   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4164   \stex_deactivate_macro:Nn \symdecl {module~environments}
4165   \stex_deactivate_macro:Nn \symdef {module~environments}
4166   \stex_deactivate_macro:Nn \notation {module~environments}
4167   \stex_reactivate_macro:N \assign
4168   \stex_reactivate_macro:N \renamedekl
4169   \stex_reactivate_macro:N \donotcopy
4170   \stex_smsmode_do:
4171 }{
4172   \stex_copymodule_end:n {
4173     \tl_if_exist:cF {
4174       l__stex_copymodule_copymodule_##1?##2_def_tl
4175     }{
4176       \str_if_eq:eeF {
4177         \prop_item:cn{
4178           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4179         }{ true }{
4180           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4181             ##1?##2
4182           }{\l_stex_current_copymodule_name_str}
4183         }
4184       }
4185     }

```

```

4186 }
4187
4188 \iffalse \begin{stex_annotate_env} \fi
4189 \NewDocumentEnvironment {realization} { 0 } { m } {
4190   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4191   \stex_deactivate_macro:Nn \symdecl {module~environments}
4192   \stex_deactivate_macro:Nn \symdef {module~environments}
4193   \stex_deactivate_macro:Nn \notation {module~environments}
4194   \stex_reactivate_macro:N \donotcopy
4195   \stex_reactivate_macro:N \assign
4196   \stex_smsmode_do:
4197 } {
4198   \stex_import_module_uri:nn { #1 } { #2 }
4199   \tl_clear:N \__stex_copymodule_exec_tl
4200   \tl_set:Nx \__stex_copymodule_module_tl {
4201     \stex_import_require_module:nnnn
4202     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4203     { \l_stex_import_path_str } { \l_stex_import_name_str }
4204   }
4205
4206   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4207     \seq_map_inline:cn {c_stex_module_##1_constants}{
4208       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4209       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4210         \stex_if_do_html:T {
4211           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4212             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4213               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__stex_
4214             }
4215           }
4216         }
4217         \tl_put_right:Nx \__stex_copymodule_module_tl {
4218           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4219         }
4220       }
4221     }
4222
4223     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4224
4225     \__stex_copymodule_exec_tl
4226     \stex_if_do_html:T {\end{stex_annotate_env}}
4227   }
4228
4229 \NewDocumentCommand \donotcopy { m } {
4230   \str_clear:N \l_stex_import_name_str
4231   \str_set:Nn \l_tmpa_str { #1 }
4232   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4233   \seq_map_inline:Nn \l_stex_all_modules_seq {
4234     \str_set:Nn \l_tmpb_str { ##1 }
4235     \str_if_eq:eeT { \l_tmpa_str } {
4236       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4237     } {
4238       \seq_map_break:n {
4239         \stex_if_do_html:T {

```

```

4240         \stex_if_smsmode:F {
4241             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4242                 \stex_annotate:nnn{domain}{##1}{}}
4243         }
4244     }
4245 }
4246 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4247 }
4248 }
4249 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4250     \str_set:Nn \l_tmpb_str { #####1 }
4251     \str_if_eq:eeT { \l_tmpa_str } {
4252         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4253     } {
4254         \seq_map_break:n {\seq_map_break:n {
4255             \stex_if_do_html:T {
4256                 \stex_if_smsmode:F {
4257                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4258                         \stex_annotate:nnn{domain}{
4259                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4260                         }}
4261                 }
4262             }
4263         }
4264         \str_set:Nx \l_stex_import_name_str {
4265             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4266         }
4267     }}
4268 }
4269 }
4270 }
4271 \str_if_empty:NTF \l_stex_import_name_str {
4272     % TODO throw error
4273 }{
4274     \stex_collect_imports:n {\l_stex_import_name_str }
4275     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4276         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4277         \seq_map_inline:cn {c_stex_module_###1_constants}{
4278             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4279             \bool_lazy_any:nT {
4280                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4281                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4282                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4283             }{
4284                 % TODO throw error
4285             }
4286         }
4287     }
4288     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4289     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4290     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4291 }
4292 \stex_smsmode_do:
4293 }

```



```

4294
4295 \NewDocumentCommand \assign { m m }{
4296   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4297   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4298   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4299   \stex_smsmode_do:
4300 }
4301
4302 \keys_define:nn { stex / renamedecl } {
4303   name          .str_set_x:N = \l_stex_renamedecl_name_str
4304 }
4305 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4306   \str_clear:N \l_stex_renamedecl_name_str
4307   \keys_set:nn { stex / renamedecl } { #1 }
4308 }
4309
4310 \NewDocumentCommand \renamedecl { O{} m m }{
4311   \__stex_copymodule_renamedecl_args:n { #1 }
4312   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4313   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4314   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4315   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4316     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4317       \l_stex_get_symbol_uri_str
4318     } }
4319   } {
4320     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4321       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4322       \prop_set_eq:cc {l_stex_symdecl_
4323         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4324         _prop
4325       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4326       \seq_set_eq:cc {l_stex_symdecl_
4327         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4328         _notations
4329       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4330       \prop_put:cnx {l_stex_symdecl_
4331         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4332         _prop
4333       }{ name }{ \l_stex_renamedecl_name_str }
4334       \prop_put:cnx {l_stex_symdecl_
4335         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4336         _prop
4337       }{ module }{ \l_stex_current_module_str }
4338       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4339         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4340       }
4341       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4342         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4343       } }
4344     }
4345   \stex_smsmode_do:
4346 }
4347

```

```

4348 \stex_deactivate_macro:Nn \assign {copymodules}
4349 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4350 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4351
4352

```

31.2 The feature environment

structural@feature

```

4353 <@@=stex_features>
4354
4355 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4356   \stex_if_in_module:F {
4357     \msg_set:nnn{stex}{error/nomodule}{
4358       Structural~Feature~has~to~occur~in~a~module:\\
4359       Feature~#2~of~type~#1\\
4360       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4361     }
4362     \msg_error:nn{stex}{error/nomodule}
4363   }
4364
4365   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4366
4367   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4368
4369   \stex_if_do_html:T {
4370     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4371     \stex_annotate_invisible:nnn{header}{\{ #3 }
4372   }
4373   }{
4374     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4375     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4376     \stex_debug:nn{features}{
4377       Feature: \l_stex_last_feature_str
4378     }
4379     \stex_if_do_html:T {
4380       \end{stex_annotate_env}
4381     }
4382   }

```

31.3 Structure

structure

```

4383 <@@=stex_structures>
4384 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4385   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4386     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4387   }
4388   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4389   {#1}{#2}
4390 }
4391

```

```

4392 \keys_define:nn { stex / features / structure } {
4393   name          .str_set_x:N = \l__stex_structures_name_str ,
4394 }
4395
4396 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4397   \str_clear:N \l__stex_structures_name_str
4398   \keys_set:nn { stex / features / structure } { #1 }
4399 }
4400
4401 \NewDocumentEnvironment{mathstructure}{m O{}}{
4402   \__stex_structures_structure_args:n { #2 }
4403   \str_if_empty:NT \l__stex_structures_name_str {
4404     \str_set:Nx \l__stex_structures_name_str { #1 }
4405   }
4406   \stex_suppress_html:n {
4407     \exp_args:Nx \stex_symdecl_do:nn {
4408       name = \l__stex_structures_name_str ,
4409       def = {\STEXsymbol{module-type}}{
4410         \stex_term_math_oms:nnnn {
4411           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4412             { ns } ?
4413           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4414             { name } / \l__stex_structures_name_str - structure
4415         }{}{0}{}
4416       }}
4417     }{ #1 }
4418   }
4419   \exp_args:Nnnx
4420   \begin{structural_feature_module}{ structure }
4421     { \l__stex_structures_name_str }{}
4422   \stex_smsmode_do:
4423 }{
4424   \end{structural_feature_module}
4425   \stex_reset_up_to_module:n \l_stex_last_feature_str
4426   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4427   \seq_clear:N \l_tmpa_seq
4428   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4429     \seq_map_inline:cn{c_stex_module_##1_constants}{
4430       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4431     }
4432   }
4433   \exp_args:Nnno
4434   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4435   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4436   \stex_add_structure_to_current_module:nn
4437     \l__stex_structures_name_str
4438     \l_stex_last_feature_str
4439
4440   \stex_execute_in_module:x {
4441     \tl_set:cn { #1 }{
4442       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4443     }
4444   }
4445 }

```

```

4446
4447 \cs_new:Nn \stex_invoke_structure:nn {
4448   \stex_invoke_symbol:n { #1?#2 }
4449 }
4450
4451 \cs_new_protected:Nn \stex_get_structure:n {
4452   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4453     \tl_set:Nn \l_tmpa_tl { #1 }
4454     \__stex_structures_get_from_cs:
4455   }{
4456     \cs_if_exist:cTF { #1 }{
4457       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4458       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4459       \str_if_empty:NNTF \l_tmpa_str {
4460         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4461           \__stex_structures_get_from_cs:
4462         }{
4463           \__stex_structures_get_from_string:n { #1 }
4464         }
4465       }{
4466         \__stex_structures_get_from_string:n { #1 }
4467       }
4468     }{
4469       \__stex_structures_get_from_string:n { #1 }
4470     }
4471   }
4472 }
4473
4474 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4475   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4476     { \tl_tail:N \l_tmpa_tl }
4477   \str_set:Nx \l_tmpa_str {
4478     \exp_after:wN \use_i:nn \l_tmpa_tl
4479   }
4480   \str_set:Nx \l_tmpb_str {
4481     \exp_after:wN \use_ii:nn \l_tmpa_tl
4482   }
4483   \str_set:Nx \l_stex_get_structure_str {
4484     \l_tmpa_str ? \l_tmpb_str
4485   }
4486   \str_set:Nx \l_stex_get_structure_module_str {
4487     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4488   }
4489 }
4490
4491 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4492   \tl_set:Nn \l_tmpa_tl {
4493     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4494   }
4495   \str_set:Nn \l_tmpa_str { #1 }
4496   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4497
4498   \seq_map_inline:Nn \l_stex_all_modules_seq {
4499     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4500 \prop_map_inline:cn {c_stex_module_##1_structures} {
4501   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4502     \prop_map_break:n{\seq_map_break:n{
4503       \tl_set:Nn \l_tmpa_tl {
4504         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4505         \str_set:Nn \l_stex_get_structure_module_str {####2}
4506       }
4507     }}
4508   }
4509 }
4510 }
4511 }
4512 \l_tmpa_tl
4513 }

```

\instantiate

```

4514
4515 \keys_define:nn { stex / instantiate } {
4516   name          .str_set_x:N = \l__stex_structures_name_str
4517 }
4518 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4519   \str_clear:N \l__stex_structures_name_str
4520   \keys_set:nn { stex / instantiate } { #1 }
4521 }
4522
4523 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4524   \beginngroup
4525     \stex_get_structure:n {#3}
4526     \__stex_structures_instantiate_args:n { #2 }
4527     \str_if_empty:NT \l__stex_structures_name_str {
4528       \str_set:Nn \l__stex_structures_name_str { #1 }
4529     }
4530     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4531     \seq_clear:N \l__stex_structures_fields_seq
4532     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4533     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4534       \seq_map_inline:cn {c_stex_module_##1_constants}{
4535         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4536       }
4537     }
4538
4539     \tl_if_empty:nF{#5}{
4540       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4541       \prop_clear:N \l_tmpa_prop
4542       \seq_map_inline:Nn \l_tmpa_seq {
4543         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4544         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4545           \msg_error:nnn{stex}{error/keyval}{##1}
4546         }
4547         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4548         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4549         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4550         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4551         \exp_args:Nxx \str_if_eq:nnF

```

```

4552         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4553         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}{
4554         \msg_error:nnxxxx{stex}{error/incompatible}
4555         {l\_stex_structures_dom_str}
4556         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4557         {\l_stex_get_symbol_uri_str}
4558         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}
4559     }
4560     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4561 }
4562 }
4563
4564 \seq_map_inline:Nn \l\_stex_structures_fields_seq {
4565     \str_set:Nx \l_tmpa_str {field:l\_stex_structures_name_str . \prop_item:cn {l_stex_sy
4566     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4567
4568     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4569     \stex_execute_in_module:x {
4570         \prop_set_from_keyval:cn { l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str_p
4571         name = \l_tmpa_str ,
4572         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4573         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4574         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4575     }
4576     \seq_clear:c {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notations}
4577 }
4578
4579 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4580     \stex_find_notation:nn{##1}{}
4581     \stex_execute_in_module:x {
4582         \seq_put_right:cn {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notation
4583     }
4584
4585     \stex_copy_control_sequence_ii:ccN
4586     {stex_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4587     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4588     \l_tmpa_tl
4589     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4590
4591
4592     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4593         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4594         \stex_execute_in_module:x {
4595             \tl_set:cn
4596             {stex_op_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4597             { \exp_args:No \exp_not:n \l_tmpa_cs}
4598         }
4599     }
4600 }
4601
4602
4603 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4604 }
4605

```

```

4606 \stex_execute_in_module:x {
4607   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4608   domain = \l_stex_get_structure_module_str ,
4609   \prop_to_keyval:N \l_tmpa_prop
4610 }
4611 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4612 }
4613 \stex_debug:nn{instantiate}{
4614   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4615   \prop_to_keyval:N \l_tmpa_prop
4616 }
4617 \exp_args:Nxx \stex_symdecl_do:nn {
4618   type={\STEXsymbol{module-type}}{
4619     \stex_term_math_oms:nnnn {
4620       \l_stex_get_structure_module_str
4621     }{}{0}{}
4622   }}
4623 }{\l__stex_structures_name_str}
4624 % {
4625   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4626   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4627   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4628 % }
4629 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4630 \endgroup
4631 \stex_smsmode_do:\ignorespacesandpars
4632 }
4633
4634 \cs_new_protected:Nn \stex_symbol_or_var:n {
4635   \cs_if_exist:cTF{#1}{
4636     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4637     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4638     \str_if_empty:NTF \l_tmpa_str {
4639       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4640       \stex_invoke_variable:n {
4641         \bool_set_true:N \l_stex_symbol_or_var_bool
4642         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4643         \str_set:Nx \l_stex_get_symbol_uri_str {
4644           \exp_after:wN \use:n \l_tmpa_tl
4645         }
4646       }{
4647         \bool_set_false:N \l_stex_symbol_or_var_bool
4648         \stex_get_symbol:n{#1}
4649       }
4650     }{
4651       \__stex_structures_symbolorvar_from_string:n{ #1 }
4652     }
4653   }{
4654     \__stex_structures_symbolorvar_from_string:n{ #1 }
4655   }
4656 }
4657
4658 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4659   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4660     \bool_set_true:N \l_stex_symbol_or_var_bool
4661     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4662   }{
4663     \bool_set_false:N \l_stex_symbol_or_var_bool
4664     \stex_get_symbol:n{#1}
4665   }
4666 }
4667
4668 \keys_define:nn { stex / varinstantiate } {
4669   name      .str_set_x:N = \l__stex_structures_name_str,
4670   bind      .choices:nn =
4671     {forall,exists}
4672     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4673 }
4674
4675 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4676   \str_clear:N \l__stex_structures_name_str
4677   \str_clear:N \l__stex_structures_bind_str
4678   \keys_set:nn { stex / varinstantiate } { #1 }
4679 }
4680
4681 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4682   \begingroup
4683     \stex_get_structure:n {#3}
4684     \__stex_structures_varinstantiate_args:n { #2 }
4685     \str_if_empty:NT \l__stex_structures_name_str {
4686       \str_set:Nn \l__stex_structures_name_str { #1 }
4687     }
4688     \stex_if_do_html:TF{
4689       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4690     }{\use:n}
4691     {
4692       \stex_if_do_html:T{
4693         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4694       }
4695       \seq_clear:N \l__stex_structures_fields_seq
4696       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4697       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4698         \seq_map_inline:cn {c_stex_module_##1_constants}{
4699           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4700         }
4701       }
4702       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4703       \prop_clear:N \l_tmpa_prop
4704       \tl_if_empty:nF {#5} {
4705         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4706         \seq_map_inline:Nn \l_tmpa_seq {
4707           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4708           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4709             \msg_error:nnn{stex}{error/keyval}{##1}
4710           }
4711           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4712           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4713           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```



```

4714 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4715 \stex_if_do_html:T{
4716   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4717 }
4718 \bool_if:NTF \l_stex_symbol_or_var_bool {
4719   \exp_args:Nxx \str_if_eq:nnF
4720     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4721     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4722     \msg_error:nnxxxx{stex}{error/incompatible}
4723     {\l__stex_structures_dom_str}
4724     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4725     {\l_stex_get_symbol_uri_str}
4726     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4727   }
4728   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4729 }{
4730   \exp_args:Nxx \str_if_eq:nnF
4731     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4732     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4733     \msg_error:nnxxxx{stex}{error/incompatible}
4734     {\l__stex_structures_dom_str}
4735     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4736     {\l_stex_get_symbol_uri_str}
4737     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4738   }
4739   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4740 }
4741 }
4742 }
4743 \tl_gclear:N \g__stex_structures_aftergroup_tl
4744 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4745   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq}{args}}
4746   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4747   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4748     \stex_find_notation:nn{##1}{
4749       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4750         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4751       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str_cs}}
4752       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4753         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4754         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4755         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str_cs}}
4756       }
4757     }
4758   }
4759   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4760     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4761       name = \l_tmpa_str ,
4762       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4763       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4764       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4765     }
4766     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4767     {g__stex_structures_tmpa_\l_tmpa_str_cs}

```

```

4768         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4769         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4770     }
4771     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4772 }
4773 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4774     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4775         domain = \l_stex_get_structure_module_str ,
4776         \prop_to_keyval:N \l_tmpa_prop
4777     }
4778     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4779     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4780         \exp_args:Nnx \exp_not:N \use:nn {
4781             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4782             \stex_term_omv:nn {var://\l__stex_structures_name_str}{
4783                 \exp_not:n{
4784                     \_varcomp{#4}
4785                 }
4786             }
4787         }{
4788             \exp_not:n{\stex_reset:N \l_stex_current_symbol_str}
4789         }
4790     }
4791 }
4792 }
4793 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4794 \aftergroup\g__stex_structures_aftergroup_tl
4795 \endgroup
4796 \stex_smsmode_do:\ignorespacesandpars
4797 }
4798
4799 \cs_new_protected:Nn \stex_invoke_instance:n {
4800     \peek_charcode_remove:NTF ! {
4801         \stex_invoke_symbol:n{#1}
4802     }{
4803         \stex_invoke_instance:nn {#1}
4804     }
4805 }
4806
4807
4808 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4809     \peek_charcode_remove:NTF ! {
4810         \exp_args:Nnx \use:nn {
4811             \def\comp{\_varcomp}
4812             \use:c{l_stex_varinstance_#1_op_tl}
4813         }{
4814             \stex_reset:N \comp
4815         }
4816     }{
4817         \stex_invoke_varinstance:nn {#1}
4818     }
4819 }
4820
4821 \cs_new_protected:Nn \stex_invoke_instance:nn {

```

```

4822 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4823   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4824 }{
4825   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4826   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4827     \prop_to_keyval:N \l_tmpa_prop
4828   }
4829 }
4830 }
4831
4832 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4833   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4834     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4835     \l_tmpa_tl
4836   }{
4837     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4838   }
4839 }

```

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

`\stex_invoke_structure:nnn`

```

4840 % #1: URI of the instance
4841 % #2: URI of the instantiated module
4842 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4843   \tl_if_empty:nTF{ #3 }{
4844     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4845       c_stex_feature_ #2 _prop
4846     }
4847     \tl_clear:N \l_tmpa_tl
4848     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4849     \seq_map_inline:Nn \l_tmpa_seq {
4850       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4851       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4852       \cs_if_exist:cT {
4853         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4854       }{
4855         \tl_if_empty:NF \l_tmpa_tl {
4856           \tl_put_right:Nn \l_tmpa_tl {,}
4857         }
4858         \tl_put_right:Nx \l_tmpa_tl {
4859           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4860         }
4861       }
4862     }
4863     \exp_args:No \mathstruct \l_tmpa_tl
4864   }{
4865     \stex_invoke_symbol:n{#1/#3}
4866   }
4867 }

```

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

```

4868 \</package>

```

Chapter 32

STEX -Statements Implementation

```
4869 <*package>
4870
4871 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4872
4873 <@@=stex_statements>
    Warnings and error messages
4874
\titleemph
4875 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4876 \keys_define:nn {stex / definiendum }{
4877   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4878   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4879   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4880   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4881 }
4882 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4883   \str_clear:N \l__stex_statements_definiendum_root_str
4884   \tl_clear:N \l__stex_statements_definiendum_post_tl
4885   \str_clear:N \l__stex_statements_definiendum_gfa_str
4886   \keys_set:nn { stex / definiendum }{ #1 }
4887 }
4888 \NewDocumentCommand \definiendum { O{} m m } {
4889   \__stex_statements_definiendum_args:n { #1 }
4890   \stex_get_symbol:n { #2 }
4891   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4892   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4893     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4894     \tl_set:Nn \l_tmpa_tl { #3 }
4895   } {
4896     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4897     \tl_set:Nn \l_tmpa_tl {
4898       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4899     }
4900   }
4901 } {
4902   \tl_set:Nn \l_tmpa_tl { #3 }
4903 }
4904
4905 % TODO root
4906 \stex_html_backend:TF {
4907   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4908 } {
4909   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4910 }
4911 }
4912 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4913
4914 \NewDocumentCommand \definame { 0{ } m } {
4915   \__stex_statements_definiendum_args:n { #1 }
4916   % TODO: root
4917   \stex_get_symbol:n { #2 }
4918   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4919   \str_set:Nx \l_tmpa_str {
4920     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4921   }
4922   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4923   \stex_html_backend:TF {
4924     \stex_if_do_html:T {
4925       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4926         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4927       }
4928     }
4929   } {
4930     \exp_args:Nnx \defemph@uri {
4931       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4932     } { \l_stex_get_symbol_uri_str }
4933   }
4934 }
4935 \stex_deactivate_macro:Nn \definame {definition~environments}
4936
4937 \NewDocumentCommand \Definame { 0{ } m } {
4938   \__stex_statements_definiendum_args:n { #1 }
4939   \stex_get_symbol:n { #2 }
4940   \str_set:Nx \l_tmpa_str {
4941     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4942   }
4943   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

4944 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4945 \stex_html_backend:TF {
4946   \stex_if_do_html:T {
4947     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4948       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4949     }
4950   }
4951 } {
4952   \exp_args:Nnx \defemph@uri {
4953     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4954   } { \l_stex_get_symbol_uri_str }
4955 }
4956 }
4957 \stex_deactivate_macro:Nn \Definame {definition-environments}
4958
4959 \NewDocumentCommand \premise { m }{
4960   \stex_annotate:nnn{ premise }{}{ #1 }
4961 }
4962 \NewDocumentCommand \conclusion { m }{
4963   \stex_annotate:nnn{ conclusion }{}{ #1 }
4964 }
4965 \NewDocumentCommand \definiens { 0{} m }{
4966   \str_clear:N \l_stex_get_symbol_uri_str
4967   \tl_if_empty:nF {#1} {
4968     \stex_get_symbol:n { #1 }
4969   }
4970   \str_if_empty:NT \l_stex_get_symbol_uri_str {
4971     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4972       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4973     }{
4974       % TODO throw error
4975     }
4976   }
4977   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
4978     {\l_stex_current_module_str}{
4979       \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4980     }{true}{
4981       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4982       \exp_args:Nx \stex_add_to_current_module:n {
4983         \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4984       }
4985     }
4986   }
4987   \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4988 }
4989
4990 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4991 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4992 \stex_deactivate_macro:Nn \definiens {definition~environments}
4993

```

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

`sdefinition`

```

4994
4995 \keys_define:nn {stex / sdefinition }{
4996   type      .str_set_x:N = \sdefinitiontype,
4997   id        .str_set_x:N = \sdefinitionid,
4998   name      .str_set_x:N = \sdefinitionname,
4999   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5000   title     .tl_set:N     = \sdefinitiontitle
5001 }
5002 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5003   \str_clear:N \sdefinitiontype
5004   \str_clear:N \sdefinitionid
5005   \str_clear:N \sdefinitionname
5006   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5007   \tl_clear:N \sdefinitiontitle
5008   \keys_set:nn { stex / sdefinition }{ #1 }
5009 }
5010
5011 \NewDocumentEnvironment{sdefinition}{O{}}{
5012   \__stex_statements_sdefinition_args:n{ #1 }
5013   \stex_reactivate_macro:N \definiendum
5014   \stex_reactivate_macro:N \definame
5015   \stex_reactivate_macro:N \Definame
5016   \stex_reactivate_macro:N \premise
5017   \stex_reactivate_macro:N \definiens
5018   \stex_if_smsmode:F{
5019     \seq_clear:N \l_tmpa_seq
5020     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5021       \tl_if_empty:nF{ ##1 }{
5022         \stex_get_symbol:n { ##1 }
5023         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5024           \l_stex_get_symbol_uri_str
5025         }
5026       }
5027     }
5028     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5029     \exp_args:Nnnx
5030     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {},}}
5031     \str_if_empty:NF \sdefinitiontype {
5032       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5033     }
5034     \str_if_empty:NF \sdefinitionname {
5035       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5036     }
5037     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5038     \tl_clear:N \l_tmpa_tl
5039     \clist_map_inline:Nn \l_tmpa_clist {
5040       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5041         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5042       }
5043     }
5044     \tl_if_empty:NTF \l_tmpa_tl {
5045       \__stex_statements_sdefinition_start:
5046     }{
5047       \l_tmpa_tl

```

```

5048     }
5049   }
5050   \stex_ref_new_doc_target:n \sdefinitionid
5051   \stex_smsmode_do:
5052 }{
5053   \stex_suppress_html:n {
5054     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5055   }
5056   \stex_if_smsmode:F {
5057     \clist_set:No \l_tmpa_clist \sdefinitiontype
5058     \tl_clear:N \l_tmpa_tl
5059     \clist_map_inline:Nn \l_tmpa_clist {
5060       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5061         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5062       }
5063     }
5064     \tl_if_empty:NTF \l_tmpa_tl {
5065       \__stex_statements_sdefinition_end:
5066     }{
5067       \l_tmpa_tl
5068     }
5069     \end{stex_annotate_env}
5070   }
5071 }

```

\stexpatchdefinition

```

5072 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5073   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5074     ~(\sdefinitiontitle)
5075   }~}
5076 }
5077 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5078
5079 \newcommand\stexpatchdefinition[3] [] {
5080   \str_set:Nx \l_tmpa_str{ #1 }
5081   \str_if_empty:NTF \l_tmpa_str {
5082     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5083     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5084   }{
5085     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5086     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5087   }
5088 }

```

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

\inlinedef inline:

```

5089 \keys_define:nn {stex / inlinedef }{
5090   type      .str_set_x:N = \sdefinitiontype,
5091   id        .str_set_x:N = \sdefinitionid,
5092   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5093   name      .str_set_x:N = \sdefinitionname
5094 }
5095 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```



```

5096 \str_clear:N \sdefinitiontype
5097 \str_clear:N \sdefinitionid
5098 \str_clear:N \sdefinitionname
5099 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5100 \keys_set:nn { stex / inlinedef }{ #1 }
5101 }
5102 \NewDocumentCommand \inlinedef { 0{} m } {
5103 \begingroup
5104 \__stex_statements_inlinedef_args:n{ #1 }
5105 \stex_reactivate_macro:N \definiendum
5106 \stex_reactivate_macro:N \definame
5107 \stex_reactivate_macro:N \Definame
5108 \stex_reactivate_macro:N \premise
5109 \stex_reactivate_macro:N \definiens
5110 \stex_ref_new_doc_target:n \sdefinitionid
5111 \stex_if_smsmode:TF{\stex_suppress_html:n {
5112 \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5113 }}{
5114 \seq_clear:N \l_tmpa_seq
5115 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5116 \tl_if_empty:nF{ ##1 }{
5117 \stex_get_symbol:n { ##1 }
5118 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5119 \l_stex_get_symbol_uri_str
5120 }
5121 }
5122 }
5123 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5124 \exp_args:Nnx
5125 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5126 \str_if_empty:NF \sdefinitiontype {
5127 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5128 }
5129 #2
5130 \str_if_empty:NF \sdefinitionname {
5131 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5132 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5133 }
5134 }
5135 }
5136 \endgroup
5137 \stex_smsmode_do:
5138 }

```

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

32.2 Assertions

sassertion

```

5139
5140 \keys_define:nn {stex / sassertion }{
5141 type .str_set_x:N = \sassertiontype,
5142 id .str_set_x:N = \sassertionid,

```

```

5143 title .tl_set:N = \sassertiontitle ,
5144 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5145 name .str_set_x:N = \sassertionname
5146 }
5147 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
5148 \str_clear:N \sassertiontype
5149 \str_clear:N \sassertionid
5150 \str_clear:N \sassertionname
5151 \clist_clear:N \l__stex_statements_sassertion_for_clist
5152 \tl_clear:N \sassertiontitle
5153 \keys_set:nn { stex / sassertion }{ #1 }
5154 }
5155
5156 %\tl_new:N \g__stex_statements_aftergroup_tl
5157
5158 \NewDocumentEnvironment{sassertion}{0{}}{
5159 \l__stex_statements_sassertion_args:n{ #1 }
5160 \stex_reactivate_macro:N \premise
5161 \stex_reactivate_macro:N \conclusion
5162 \stex_if_smsmode:F {
5163 \seq_clear:N \l_tmpa_seq
5164 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5165 \tl_if_empty:nF{ ##1 }{
5166 \stex_get_symbol:n { ##1 }
5167 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5168 \l_stex_get_symbol_uri_str
5169 }
5170 }
5171 }
5172 \exp_args:Nnnx
5173 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5174 \str_if_empty:NF \sassertiontype {
5175 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5176 }
5177 \str_if_empty:NF \sassertionname {
5178 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5179 }
5180 \clist_set:Nn \l_tmpa_clist \sassertiontype
5181 \tl_clear:N \l_tmpa_tl
5182 \clist_map_inline:Nn \l_tmpa_clist {
5183 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5184 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5185 }
5186 }
5187 \tl_if_empty:NTF \l_tmpa_tl {
5188 \l__stex_statements_sassertion_start:
5189 }{
5190 \l_tmpa_tl
5191 }
5192 }
5193 \str_if_empty:NTF \sassertionid {
5194 \str_if_empty:NF \sassertionname {
5195 \stex_ref_new_doc_target:n {}
5196 }

```

```

5197 } {
5198   \stex_ref_new_doc_target:n \sassertionid
5199 }
5200 \stex_smsmode_do:
5201 }{
5202   \str_if_empty:NF \sassertionname {
5203     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5204     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5205   }
5206   \stex_if_smsmode:F {
5207     \clist_set:N \l_tmpa_clist \sassertiontype
5208     \tl_clear:N \l_tmpa_tl
5209     \clist_map_inline:Nn \l_tmpa_clist {
5210       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5211         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5212       }
5213     }
5214     \tl_if_empty:NTF \l_tmpa_tl {
5215       __stex_statements_sassertion_end:
5216     }{
5217       \l_tmpa_tl
5218     }
5219     \end{stex_annotate_env}
5220   }
5221 }

```

\stexpatchassertion

```

5222
5223 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5224   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5225     (\sassertiontitle)
5226   }~}
5227 }
5228 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5229
5230 \newcommand\stexpatchassertion[3] [] {
5231   \str_set:Nx \l_tmpa_str{ #1 }
5232   \str_if_empty:NTF \l_tmpa_str {
5233     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5234     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5235   }{
5236     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5237     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5238   }
5239 }

```

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

\inlineass inline:

```

5240 \keys_define:nn {stex / inlineass }{
5241   type      .str_set_x:N = \sassertiontype,
5242   id        .str_set_x:N = \sassertionid,
5243   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5244   name      .str_set_x:N = \sassertionname

```

```

5245 }
5246 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5247   \str_clear:N \sassertiontype
5248   \str_clear:N \sassertionid
5249   \str_clear:N \sassertionname
5250   \clist_clear:N \l__stex_statements_sassertion_for_clist
5251   \keys_set:nn { stex / inlineass }{ #1 }
5252 }
5253 \NewDocumentCommand \inlineass { 0{} m } {
5254   \begingroup
5255   \stex_reactivate_macro:N \premise
5256   \stex_reactivate_macro:N \conclusion
5257   \__stex_statements_inlineass_args:n{ #1 }
5258   \str_if_empty:NTF \sassertionid {
5259     \str_if_empty:NF \sassertionname {
5260       \stex_ref_new_doc_target:n {}
5261     }
5262   } {
5263     \stex_ref_new_doc_target:n \sassertionid
5264   }
5265
5266   \stex_if_smsmode:TF{
5267     \str_if_empty:NF \sassertionname {
5268       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5269       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5270     }
5271   }{
5272     \seq_clear:N \l_tmpa_seq
5273     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5274       \tl_if_empty:nF{ ##1 }{
5275         \stex_get_symbol:n { ##1 }
5276         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5277           \l_stex_get_symbol_uri_str
5278         }
5279       }
5280     }
5281     \exp_args:Nnx
5282     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5283       \str_if_empty:NF \sassertiontype {
5284         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5285       }
5286       #2
5287       \str_if_empty:NF \sassertionname {
5288         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5289         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5290         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5291       }
5292     }
5293   }
5294   \endgroup
5295   \stex_smsmode_do:
5296 }

```

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

32.3 Examples

sexample

```

5297
5298 \keys_define:nn {stex / sexample }{
5299   type      .str_set_x:N = \exampletype,
5300   id        .str_set_x:N = \sexampleid,
5301   title     .tl_set:N     = \sexampletile,
5302   name      .str_set_x:N = \sexamplename ,
5303   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5304 }
5305 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5306   \str_clear:N \sexampletype
5307   \str_clear:N \sexampleid
5308   \str_clear:N \sexamplename
5309   \tl_clear:N \sexampletile
5310   \clist_clear:N \l__stex_statements_sexample_for_clist
5311   \keys_set:nn { stex / sexample }{ #1 }
5312 }
5313
5314 \NewDocumentEnvironment{sexample}{0{}}{
5315   \__stex_statements_sexample_args:n{ #1 }
5316   \stex_reactivate_macro:N \premise
5317   \stex_reactivate_macro:N \conclusion
5318   \stex_if_smsmode:F {
5319     \seq_clear:N \l_tmpa_seq
5320     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5321       \tl_if_empty:NF{ ##1 }{
5322         \stex_get_symbol:n { ##1 }
5323         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5324           \l_stex_get_symbol_uri_str
5325         }
5326       }
5327     }
5328     \exp_args:Nnnx
5329     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5330     \str_if_empty:NF \sexampletype {
5331       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5332     }
5333     \str_if_empty:NF \sexamplename {
5334       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5335     }
5336     \clist_set:Nn \l_tmpa_clist \sexampletype
5337     \tl_clear:N \l_tmpa_tl
5338     \clist_map_inline:Nn \l_tmpa_clist {
5339       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5340         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5341       }
5342     }
5343     \tl_if_empty:NTF \l_tmpa_tl {
5344       \__stex_statements_sexample_start:
5345     }{
5346       \l_tmpa_tl
5347     }

```

```

5348 }
5349 \str_if_empty:NF \sexampleid {
5350   \stex_ref_new_doc_target:n \sexampleid
5351 }
5352 \stex_smsmode_do:
5353 ){
5354   \str_if_empty:NF \sexamplename {
5355     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5356   }
5357   \stex_if_smsmode:F {
5358     \clist_set:Nn \l_tmpa_clist \sexamplotype
5359     \tl_clear:N \l_tmpa_tl
5360     \clist_map_inline:Nn \l_tmpa_clist {
5361       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5362         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5363       }
5364     }
5365     \tl_if_empty:NTF \l_tmpa_tl {
5366       \__stex_statements_sexample_end:
5367     }{
5368       \l_tmpa_tl
5369     }
5370     \end{stex_annotate_env}
5371   }
5372 }

```

\stexpatchexample

```

5373
5374 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5375   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5376     (\sexamplotype)
5377   }~}
5378 }
5379 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5380
5381 \newcommand\stexpatchexample[3] [] {
5382   \str_set:Nx \l_tmpa_str{ #1 }
5383   \str_if_empty:NTF \l_tmpa_str {
5384     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5385     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5386   }{
5387     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5388     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5389   }
5390 }

```

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

\inlineex inline:

```

5391 \keys_define:nn {stex / inlineex }{
5392   type      .str_set_x:N = \sexamplotype,
5393   id        .str_set_x:N = \sexampleid,
5394   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5395   name      .str_set_x:N = \sexamplename

```

```

5396 }
5397 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5398   \str_clear:N \sexamplotype
5399   \str_clear:N \sexampleid
5400   \str_clear:N \sexamplename
5401   \clist_clear:N \l__stex_statements_sexample_for_clist
5402   \keys_set:nn { stex / inlineex }{ #1 }
5403 }
5404 \NewDocumentCommand \inlineex { 0{ } m } {
5405   \beginngroup
5406   \stex_reactivate_macro:N \premise
5407   \stex_reactivate_macro:N \conclusion
5408   \__stex_statements_inlineex_args:n{ #1 }
5409   \str_if_empty:NF \sexampleid {
5410     \stex_ref_new_doc_target:n \sexampleid
5411   }
5412   \stex_if_smsmode:TF{
5413     \str_if_empty:NF \sexamplename {
5414       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5415     }
5416   }{
5417     \seq_clear:N \l_tmpa_seq
5418     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5419       \tl_if_empty:nF{ ##1 }{
5420         \stex_get_symbol:n { ##1 }
5421         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5422           \l_stex_get_symbol_uri_str
5423         }
5424       }
5425     }
5426     \exp_args:Nnx
5427     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5428       \str_if_empty:NF \sexamplotype {
5429         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5430       }
5431       #2
5432       \str_if_empty:NF \sexamplename {
5433         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5434         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5435       }
5436     }
5437   }
5438   \endgroup
5439   \stex_smsmode_do:
5440 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5441 \keys_define:nn { stex / sparagraph } {
5442   id          .str_set_x:N    = \sparagraphid ,

```

```

5443 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5444 type .str_set_x:N = \sparagraphtype ,
5445 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5446 from .tl_set:N = \sparagraphfrom ,
5447 to .tl_set:N = \sparagraphto ,
5448 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5449 name .str_set:N = \sparagraphname ,
5450 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5451 }
5452
5453 \cs_new_protected:Nn \stex_sparagraph_args:n {
5454 \tl_clear:N \l_stex_sparagraph_title_tl
5455 \tl_clear:N \sparagraphfrom
5456 \tl_clear:N \sparagraphto
5457 \tl_clear:N \l_stex_sparagraph_start_tl
5458 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5459 \str_clear:N \sparagraphid
5460 \str_clear:N \sparagraphtype
5461 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5462 \str_clear:N \sparagraphname
5463 \keys_set:nn { stex / sparagraph }{ #1 }
5464 }
5465 \newif\if@in@omtext\@in@omtextfalse
5466
5467 \NewDocumentEnvironment {sparagraph} { 0{} } {
5468 \stex_sparagraph_args:n { #1 }
5469 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5470 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5471 }{
5472 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5473 }
5474 \@in@omtexttrue
5475 \stex_if_smsmode:F {
5476 \seq_clear:N \l_tmpa_seq
5477 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5478 \tl_if_empty:NF{ ##1 }{
5479 \stex_get_symbol:n { ##1 }
5480 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5481 \l_stex_get_symbol_uri_str
5482 }
5483 }
5484 }
5485 \exp_args:Nnnx
5486 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5487 \str_if_empty:NF \sparagraphtype {
5488 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5489 }
5490 \str_if_empty:NF \sparagraphfrom {
5491 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5492 }
5493 \str_if_empty:NF \sparagraphto {
5494 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5495 }
5496 \str_if_empty:NF \sparagraphname {

```



```

5497     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5498   }
5499   \clist_set:No \l_tmpa_clist \sparagraphtype
5500   \tl_clear:N \l_tmpa_tl
5501   \clist_map_inline:Nn \sparagraphtype {
5502     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5503       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5504     }
5505   }
5506   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5507   \tl_if_empty:NTF \l_tmpa_tl {
5508     \__stex_statements_sparagraph_start:
5509   }{
5510     \l_tmpa_tl
5511   }
5512 }
5513 \clist_set:No \l_tmpa_clist \sparagraphtype
5514 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5515 {
5516   \stex_reactivate_macro:N \definiendum
5517   \stex_reactivate_macro:N \definame
5518   \stex_reactivate_macro:N \Definame
5519   \stex_reactivate_macro:N \premise
5520   \stex_reactivate_macro:N \definiens
5521 }
5522 \str_if_empty:NTF \sparagraphid {
5523   \str_if_empty:NTF \sparagraphname {
5524     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5525       \stex_ref_new_doc_target:n {}
5526     }
5527   } {
5528     \stex_ref_new_doc_target:n {}
5529   }
5530 } {
5531   \stex_ref_new_doc_target:n \sparagraphid
5532 }
5533 \exp_args:NNx
5534 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5535   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5536     \tl_if_empty:nF{ ##1 }{
5537       \stex_get_symbol:n { ##1 }
5538       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5539     }
5540   }
5541 }
5542 \stex_smsmode_do:
5543 \ignorespacesandpars
5544 }{
5545   \str_if_empty:NF \sparagraphname {
5546     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5547     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5548   }
5549   \stex_if_smsmode:F {
5550     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5551 \tl_clear:N \l_tmpa_tl
5552 \clist_map_inline:Nn \l_tmpa_clist {
5553   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5554     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5555   }
5556 }
5557 \tl_if_empty:NTF \l_tmpa_tl {
5558   \__stex_statements_sparagraph_end:
5559 }{
5560   \l_tmpa_tl
5561 }
5562 \end{stex_annotate_env}
5563 }
5564 }

```

\stexpatchparagraph

```

5565
5566 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5567   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5568     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5569       \titleemph{\l_stex_sparagraph_title_tl}:~
5570     }
5571   }{
5572     \titleemph{\l_stex_sparagraph_start_tl}~
5573   }
5574 }
5575 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5576
5577 \newcommand\stexpatchparagraph[3] [] {
5578   \str_set:Nx \l_tmpa_str{ #1 }
5579   \str_if_empty:NTF \l_tmpa_str {
5580     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5581     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5582   }{
5583     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5584     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5585   }
5586 }
5587
5588 \keys_define:nn { stex / inlinepara } {
5589   id      .str_set_x:N = \sparagraphid ,
5590   type    .str_set_x:N = \sparagraphtype ,
5591   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5592   from    .tl_set:N    = \sparagraphfrom ,
5593   to      .tl_set:N    = \sparagraphto ,
5594   name    .str_set:N   = \sparagraphname
5595 }
5596 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5597   \tl_clear:N \sparagraphfrom
5598   \tl_clear:N \sparagraphto
5599   \str_clear:N \sparagraphid
5600   \str_clear:N \sparagraphtype
5601   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5602   \str_clear:N \sparagraphname

```

```

5603 \keys_set:nn { stex / inlinepara }{ #1 }
5604 }
5605 \NewDocumentCommand \inlinepara { 0{} m } {
5606   \beginingroup
5607   \__stex_statements_inlinepara_args:n{ #1 }
5608   \clist_set:No \l_tmpa_clist \sparagraphtype
5609   \str_if_empty:NTF \sparagraphid {
5610     \str_if_empty:NTF \sparagraphname {
5611       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5612         \stex_ref_new_doc_target:n {}
5613       }
5614     } {
5615       \stex_ref_new_doc_target:n {}
5616     }
5617   } {
5618     \stex_ref_new_doc_target:n \sparagraphid
5619   }
5620   \stex_if_smsmode:TF{
5621     \str_if_empty:NF \sparagraphname {
5622       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5623     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5624   }
5625 }{
5626   \seq_clear:N \l_tmpa_seq
5627   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5628     \tl_if_empty:nF{ ##1 }{
5629       \stex_get_symbol:n { ##1 }
5630       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5631         \l_stex_get_symbol_uri_str
5632       }
5633     }
5634   }
5635   \exp_args:Nnx
5636   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5637     \str_if_empty:NF \sparagraphtype {
5638       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5639     }
5640     \str_if_empty:NF \sparagraphfrom {
5641       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5642     }
5643     \str_if_empty:NF \sparagraphto {
5644       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5645     }
5646     \str_if_empty:NF \sparagraphname {
5647       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5648     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5649     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5650   }
5651   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5652     \clist_map_inline:Nn \l_tmpa_seq {
5653       \stex_ref_new_sym_target:n {##1}
5654     }
5655   }
5656   #2

```

```

5657     }
5658   }
5659   \endgroup
5660   \stex_smsmode_do:
5661 }
5662
(End definition for \stexpatchparagraph. This function is documented on page 47.)
5663 \</package>

```

Chapter 33

The Implementation

```
5664 <*package>
5665 <@@=stex_sproof>
5666
5667 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5668
```

33.1 Proofs

We first define some keys for the proof environment.

```
5669 \keys_define:nn { stex / spf } {
5670   id          .str_set_x:N = \spfid,
5671   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5672   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5673   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5674   type        .str_set_x:N = \spftype,
5675   title       .tl_set:N    = \spftitle,
5676   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5677   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5678   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5679 }
5680 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5681   \str_clear:N \spfid
5682   \tl_clear:N \l__stex_sproof_spf_for_tl
5683   \tl_clear:N \l__stex_sproof_spf_from_tl
5684   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5685   \str_clear:N \spftype
5686   \tl_clear:N \spftitle
5687   \tl_clear:N \l__stex_sproof_spf_continues_tl
5688   \tl_clear:N \l__stex_sproof_spf_functions_tl
5689   \tl_clear:N \l__stex_sproof_spf_method_tl
5690   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5691   \keys_set:nn { stex / spf }{ #1 }
5692 }
```

\c__stex_sproof_flow_str

We define this macro, so that we can test whether the display key has the value flow

```
5693 \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.

```

5694 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5695 \cs_new_protected:Npn \sproofnumber {
5696   \int_set:Nn \l_tmpa_int {1}
5697   \bool_while_do:nn {
5698     \int_compare_p:nNn {
5699       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5700     } > 0
5701   }{
5702     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5703     \int_incr:N \l_tmpa_int
5704   }
5705 }
5706 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5707   \int_set:Nn \l_tmpa_int {1}
5708   \bool_while_do:nn {
5709     \int_compare_p:nNn {
5710       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5711     } > 0
5712   }{
5713     \int_incr:N \l_tmpa_int
5714   }
5715   \int_compare:nNnF \l_tmpa_int = 1 {
5716     \int_decr:N \l_tmpa_int
5717   }
5718   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5719     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5720   }
5721 }
5722
5723 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5724   \int_set:Nn \l_tmpa_int {1}
5725   \bool_while_do:nn {
5726     \int_compare_p:nNn {
5727       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5728     } > 0
5729   }{
5730     \int_incr:N \l_tmpa_int
5731   }
5732   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5733 }
5734
5735 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5736   \int_set:Nn \l_tmpa_int {1}
5737   \bool_while_do:nn {

```

```

5738 \int_compare_p:nNn {
5739 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5740 } > 0
5741 }{
5742 \int_incr:N \l_tmpa_int
5743 }
5744 \int_decr:N \l_tmpa_int
5745 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5746 }

```

\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

```

5747 \def\sproof@box{
5748 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5749 }
5750 \def\sproofend{
5751 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5752 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5753 }
5754 }

```

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

spf@*@kw

```

5755 \def\spf@proofsketch@kw{Proof~Sketch}
5756 \def\spf@proof@kw{Proof}
5757 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5758 \AddToHook{begindocument}{
5759 \ltx@ifpackageloaded{babel}{
5760 \makeatletter
5761 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5762 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5763 \input{sproof-ngerman.ldf}
5764 }
5765 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5766 \input{sproof-finnish.ldf}
5767 }
5768 \clist_if_in:NnT \l_tmpa_clist {french}{
5769 \input{sproof-french.ldf}
5770 }
5771 \clist_if_in:NnT \l_tmpa_clist {russian}{
5772 \input{sproof-russian.ldf}
5773 }
5774 \makeatother
5775 }{}
5776 }

```

spfsketch

```

5777 \newcommand\spfsketch[2][]{
5778 \begin{group}
5779 \let \premise \stex_proof_premise:

```

```

5780 \stex_sproof_spf_args:n{#1}
5781 \stex_if_smsmode:TF {
5782   \str_if_empty:NF \spfid {
5783     \stex_ref_new_doc_target:n \spfid
5784   }
5785 }{
5786   \seq_clear:N \l_tmpa_seq
5787   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5788     \tl_if_empty:nF{ ##1 }{
5789       \stex_get_symbol:n { ##1 }
5790       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5791         \l_stex_get_symbol_uri_str
5792       }
5793     }
5794   }
5795   \exp_args:Nnx
5796   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5797     \str_if_empty:NF \spftype {
5798       \stex_annotate_invisible:nnn{type}{\spftype}{
5799     }
5800     \clist_set:No \l_tmpa_clist \spftype
5801     \tl_set:Nn \l_tmpa_tl {
5802       \titleemph{
5803         \tl_if_empty:NTF \spftitle {
5804           \spf@proofsketch@kw
5805         }{
5806           \spftitle
5807         }
5808       }::~
5809     }
5810     \clist_map_inline:Nn \l_tmpa_clist {
5811       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5812         \tl_clear:N \l_tmpa_tl
5813       }
5814     }
5815     \str_if_empty:NF \spfid {
5816       \stex_ref_new_doc_target:n \spfid
5817     }
5818     \l_tmpa_tl #2 \sproofend
5819   }
5820 }
5821 \endgroup
5822 \stex_smsmode_do:
5823 }
5824

```

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

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

```

5825 \newenvironment{spfeq}[2][]{
5826   \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


```

5827 \let \premise \stex_proof_premise:
5828 \stex_if_smsmode:TF {
5829   \str_if_empty:NF \spfid {
5830     \stex_ref_new_doc_target:n \spfid
5831   }
5832 }{
5833   \seq_clear:N \l_tmpa_seq
5834   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5835     \tl_if_empty:nF{ ##1 }{
5836       \stex_get_symbol:n { ##1 }
5837       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5838         \l_stex_get_symbol_uri_str
5839       }
5840     }
5841   }
5842   \exp_args:Nnnx
5843   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5844   \str_if_empty:NF \spftype {
5845     \stex_annotate_invisible:nnn{type}{\spftype}{}
5846   }
5847
5848   \clist_set:No \l_tmpa_clist \spftype
5849   \tl_clear:N \l_tmpa_tl
5850   \clist_map_inline:Nn \l_tmpa_clist {
5851     \tl_if_exist:cT {\_stex_sproof_spfeq_##1_start:}{
5852       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_start:}}
5853     }
5854     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5855       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5856     }
5857   }
5858   \tl_if_empty:NTF \l_tmpa_tl {
5859     \_stex_sproof_spfeq_start:
5860   }{
5861     \l_tmpa_tl
5862   }{~#2}
5863   \str_if_empty:NF \spfid {
5864     \stex_ref_new_doc_target:n \spfid
5865   }
5866   \begin{displaymath}\begin{array}{rc1l}
5867   }
5868   \stex_smsmode_do:
5869 }{
5870   \stex_if_smsmode:F {
5871     \end{array}\end{displaymath}
5872     \clist_set:No \l_tmpa_clist \spftype
5873     \tl_clear:N \l_tmpa_tl
5874     \clist_map_inline:Nn \l_tmpa_clist {
5875       \tl_if_exist:cT {\_stex_sproof_spfeq_##1_end:}{
5876         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_end:}}
5877       }
5878     }
5879     \tl_if_empty:NTF \l_tmpa_tl {
5880       \_stex_sproof_spfeq_end:

```

```

5881     }{
5882       \l_tmpa_tl
5883     }
5884     \end{stex_annotate_env}
5885   }
5886 }
5887
5888 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5889   \titleemph{
5890     \tl_if_empty:NTF \spftitle {
5891       \spf@proof@kw
5892     }{
5893       \spftitle
5894     }
5895   }:
5896 }
5897 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5898
5899 \newcommand\stexpatchspfeq[3] [] {
5900   \str_set:Nx \l_tmpa_str{ #1 }
5901   \str_if_empty:NTF \l_tmpa_str {
5902     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5903     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5904   }{
5905     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5906     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5907   }
5908 }
5909

```

(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.

```

5910 \newenvironment{sproof}[2] []{
5911   \let \premise \stex_proof_premise:
5912   \intarray_gzero:N \l__stex_sproof_counter_intarray
5913   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5914   \__stex_sproof_spf_args:n{#1}
5915   \stex_if_smsmode:TF {
5916     \str_if_empty:NF \spfid {
5917       \stex_ref_new_doc_target:n \spfid
5918     }
5919   }{
5920     \seq_clear:N \l_tmpa_seq
5921     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5922       \tl_if_empty:nF{ ##1 }{
5923         \stex_get_symbol:n { ##1 }
5924         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5925           \l_stex_get_symbol_uri_str
5926         }
5927       }
5928     }

```

```

5929 \exp_args:Nnnx
5930 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5931 \str_if_empty:NF \spftype {
5932   \stex_annotate_invisible:nnn{type}{\spftype}{}
5933 }
5934
5935 \clist_set:No \l_tmpa_clist \spftype
5936 \tl_clear:N \l_tmpa_tl
5937 \clist_map_inline:Nn \l_tmpa_clist {
5938   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5939     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5940   }
5941   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5942     \tl_set:Nn \l_tmpa_tl {\use:n{}}
5943   }
5944 }
5945 \tl_if_empty:NTF \l_tmpa_tl {
5946   \__stex_sproof_sproof_start:
5947 }{
5948   \l_tmpa_tl
5949 }{~#2}
5950 \str_if_empty:NF \spfid {
5951   \stex_ref_new_doc_target:n \spfid
5952 }
5953 \begin{description}
5954 }
5955 \stex_smsmode_do:
5956 }{
5957   \stex_if_smsmode:F{
5958     \end{description}
5959     \clist_set:No \l_tmpa_clist \spftype
5960     \tl_clear:N \l_tmpa_tl
5961     \clist_map_inline:Nn \l_tmpa_clist {
5962       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5963         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5964       }
5965     }
5966     \tl_if_empty:NTF \l_tmpa_tl {
5967       \__stex_sproof_sproof_end:
5968     }{
5969       \l_tmpa_tl
5970     }
5971     \end{stex_annotate_env}
5972   }
5973 }
5974
5975 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5976   \par\noindent\titleemph{
5977     \tl_if_empty:NTF \spftype {
5978       \spf@proof@kw
5979     }{
5980       \spftype
5981     }
5982   }::

```

```

5983 }
5984 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5985
5986 \newcommand\stexpatchproof[3] [] {
5987   \str_set:Nx \l_tmpa_str{ #1 }
5988   \str_if_empty:NTF \l_tmpa_str {
5989     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5990     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5991   }{
5992     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5993     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5994   }
5995 }

```

\spfidea

```

5996 \newcommand\spfidea[2] []{
5997   \__stex_sproof_spf_args:n{#1}
5998   \titleemph{
5999     \tl_if_empty:NTF \spftype {Proof-Idea}{
6000       \spftype
6001     }:
6002   }~#2
6003   \sproofend
6004 }

```

(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

```

6005 \newenvironment{spfstep}[1] []{
6006   \__stex_sproof_spf_args:n{#1}
6007   \stex_if_smsmode:TF {
6008     \str_if_empty:NF \spfid {
6009       \stex_ref_new_doc_target:n \spfid
6010     }
6011   }{
6012     \@in@omtexttrue
6013     \seq_clear:N \l_tmpa_seq
6014     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6015       \tl_if_empty:nF{ ##1 }{
6016         \stex_get_symbol:n { ##1 }
6017         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6018           \l_stex_get_symbol_uri_str
6019         }
6020       }
6021     }
6022     \exp_args:Nnnx
6023     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6024     \str_if_empty:NF \spftype {
6025       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6026   }
6027   \clist_set:No \l_tmpa_clist \spftype
6028   \tl_set:Nn \l_tmpa_tl {
6029     \item[\sproofnumber]
6030     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6031   }
6032   \clist_map_inline:Nn \l_tmpa_clist {
6033     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6034       \tl_clear:N \l_tmpa_tl
6035     }
6036   }
6037   \l_tmpa_tl
6038   \tl_if_empty:NF \spftitle {
6039     {(\titleemph{\spftitle})\enspace}
6040   }
6041   \str_if_empty:NF \spfid {
6042     \stex_ref_new_doc_target:n \spfid
6043   }
6044 }
6045 \stex_smsmode_do:
6046 \ignorespacesandpars
6047 }{
6048   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6049     \__stex_sproof_inc_counter:
6050   }
6051   \stex_if_smsmode:F {
6052     \end{stex_annotate_env}
6053   }
6054 }

```

spfcomment

```

6055 \newenvironment{spfcomment}[1][]{
6056   \__stex_sproof_spf_args:n{#1}
6057   \clist_set:No \l_tmpa_clist \spftype
6058   \tl_set:Nn \l_tmpa_tl {
6059     \item[\sproofnumber]
6060     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6061   }
6062   \clist_map_inline:Nn \l_tmpa_clist {
6063     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6064       \tl_clear:N \l_tmpa_tl
6065     }
6066   }
6067   \l_tmpa_tl
6068 }{
6069   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6070     \__stex_sproof_inc_counter:
6071   }
6072 }

```

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.

```

6073 \newenvironment{subproof}[2][]{
6074   \_stex_sproof_spf_args:n{#1}
6075   \stex_if_smsmode:TF{
6076     \str_if_empty:NF \spfid {
6077       \stex_ref_new_doc_target:n \spfid
6078     }
6079   }{
6080     \seq_clear:N \l_tmpa_seq
6081     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6082       \tl_if_empty:nF{ ##1 }{
6083         \stex_get_symbol:n { ##1 }
6084         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6085           \l_stex_get_symbol_uri_str
6086         }
6087       }
6088     }
6089     \exp_args:Nnnx
6090     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6091     \str_if_empty:NF \spftype {
6092       \stex_annotate_invisible:nnn{type}{\spftype}{}
6093     }
6094
6095     \clist_set:No \l_tmpa_clist \spftype
6096     \tl_set:Nn \l_tmpa_tl {
6097       \item[\sproofnumber]
6098       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6099     }
6100     \clist_map_inline:Nn \l_tmpa_clist {
6101       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6102         \tl_clear:N \l_tmpa_tl
6103       }
6104     }
6105     \l_tmpa_tl
6106     \tl_if_empty:NF \spftitle {
6107       {(\titleemph{\spftitle})\enspace}
6108     }
6109     {\~#2}
6110     \str_if_empty:NF \spfid {
6111       \stex_ref_new_doc_target:n \spfid
6112     }
6113   }
6114   \_stex_sproof_add_counter:
6115   \stex_smsmode_do:
6116 }{
6117   \_stex_sproof_remove_counter:
6118   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6119     \_stex_sproof_inc_counter:
6120   }
6121   \stex_if_smsmode:F{
6122     \end{stex_annotate_env}
6123   }
6124 }

```

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

```

6125 \newenvironment{spfcases}[2] [] {
6126   \tl_if_empty:nTF{#1}{
6127     \begin{subproof}[method=by-cases]{#2}
6128   }{
6129     \begin{subproof}[#1,method=by-cases]{#2}
6130   }
6131 }{
6132   \end{subproof}
6133 }

```

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

```

6134 \newenvironment{spfcase}[2] [] {
6135   \__stex_sproof_spf_args:n{#1}
6136   \stex_if_smsmode:TF {
6137     \str_if_empty:NF \spfid {
6138       \stex_ref_new_doc_target:n \spfid
6139     }
6140   }{
6141     \seq_clear:N \l_tmpa_seq
6142     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6143       \tl_if_empty:nF{ ##1 }{
6144         \stex_get_symbol:n { ##1 }
6145         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6146           \l_stex_get_symbol_uri_str
6147         }
6148       }
6149     }
6150     \exp_args:Nnnx
6151     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6152     \str_if_empty:NF \spftype {
6153       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6154     }
6155     \clist_set:No \l_tmpa_clist \spftype
6156     \tl_set:Nn \l_tmpa_tl {
6157       \item[\sproofnumber]
6158       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6159     }
6160     \clist_map_inline:Nn \l_tmpa_clist {
6161       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6162         \tl_clear:N \l_tmpa_tl
6163       }
6164     }
6165     \l_tmpa_tl
6166     \tl_if_empty:nF{#2}{
6167       \titleemph{#2}:~
6168     }
6169   }
6170   \__stex_sproof_add_counter:
6171   \stex_smsmode_do:
6172 }{
6173   \__stex_sproof_remove_counter:
6174   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6175     \__stex_sproof_inc_counter:

```

```

6176 }
6177 \stex_if_smsmode:F{
6178   \clist_set:No \l_tmpa_clist \spftype
6179   \tl_set:Nn \l_tmpa_tl{\sproofend}
6180   \clist_map_inline:Nn \l_tmpa_clist {
6181     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6182       \tl_clear:N \l_tmpa_tl
6183     }
6184   }
6185   \l_tmpa_tl
6186   \end{stex_annotate_env}
6187 }
6188 }

```

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

```

6189 \newcommand\spfcasesketch[3] [] {
6190   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6191 }

```

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.

```

6192 \keys_define:nn { stex / just }{
6193   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6194   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6195   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6196   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6197 }

```

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

\spfjust

```

6198 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6199 \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.

```

6200 \newcommand\justarg[2] [] {#2}
6201 \end{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

```
6202 <*package>
6203
6204 %%%%%%%%%% others.dtx %%%%%%%%%%
6205
6206 <@@=stex_others>
        Warnings and error messages
6207 % None

\MSC Math subject classifier

6208 \NewDocumentCommand \MSC {m} {
6209 % TODO
6210 }

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

6211 \@ifpackageloaded{tikzinput}{
6212 \RequirePackage{stex-tikzinput}
6213 }{}
6214
6215 \bool_if:NT \c_stex_persist_mode_bool {
6216 \input{\jobname.sms}
6217 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6218 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6219 \l_tmpa_str
6220 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6221 \c_stex_mathhub_main_manifest_prop
6222 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6223 }
6224 }

6225 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6226 <*package>
6227 <@@=stex_modules>
6228
6229 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6230
6231 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6232 \begingroup
6233 \stex_module_setup:nn{
6234   ns=\c_stex_metatheory_ns_str,
6235   meta=NONE
6236 }{Metatheory}
6237 \stex_reactivate_macro:N \symdecl
6238 \stex_reactivate_macro:N \notation
6239 \stex_reactivate_macro:N \symdef
6240 \ExplSyntaxOff
6241 \csname stex_suppress_html:n\endcsname{
6242   % is-a (a:A, a \in A, a is an A, etc.)
6243   \symdecl{isa}[args=ai]
6244   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6245   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6246   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6247
6248   % bind (\forall, \Pi, \lambda etc.)
6249   \symdecl{bind}[args=Bi]
6250   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6251   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6252   \notation{bind}[depfun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
6253
6254   % implicit bind
6255   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6256
6257   % dummy variable
6258   \symdecl{dummyvar}
6259   \notation{dummyvar}[underscore]{\comp\_}
6260   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6261 \notation{dummyvar}[dash]{\comp{\rm --}}
6262
6263 %fromto (function space, Hom-set, implication etc.)
6264 \symdecl{fromto}[args=ai]
6265 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6266 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6267
6268 % mapto (lambda etc.)
6269 \symdecl{mapto}[args=Bi]
6270 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6271 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6272 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6273
6274 % function/operator application
6275 \symdecl{apply}[args=ia]
6276 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6277 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6278
6279 % collection of propositions/booleans/truth values
6280 \symdecl{prop}[name=proposition]
6281 \notation{prop}[prop]{\comp{\rm prop}}
6282 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6283
6284 \symdecl{judgmentholds}[args=1]
6285 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6286
6287 % sequences
6288 \symdecl{seqtype}[args=1]
6289 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6290
6291 \symdecl{seqexpr}[args=a]
6292 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6293
6294 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6295 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6296 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6297 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6298 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6299 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6300 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6301 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6302 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6303
6304 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6305 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6306
6307 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6308 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6309 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6310
6311 % letin ("let", local definitions, variable substitution)
6312 \symdecl{letin}[args=bii]
6313 \notation{letin}[let]{\comp{\rm let}\; ;#1\comp{=}\#2\; ;\comp{\rm in}\; ;#3}
6314 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6315 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6316
6317 % structures
6318 \symdecl*{module-type}[args=1]
6319 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6320 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6321 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6322
6323 % objects
6324 \symdecl{object}
6325 \notation{object}{\comp{\mathtt{OBJECT}}}}
6326
6327 }
6328
6329 % The following are abbreviations in the sTeX corpus that are left over from earlier
6330 % developments. They will eventually be phased out.
6331
6332 \ExplSyntaxOn
6333 \stex_add_to_current_module:n{
6334   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6335   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6336   \def\livar{\csname sequence-index\endcsname[li]}
6337   \def\uivar{\csname sequence-index\endcsname[ui]}
6338   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6339   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6340 }
6341 \__stex_modules_end_module:
6342 \endgroup
6343 \endpackage

```

Chapter 36

Tikzinput Implementation

```
6344 <@@=tikzinput>
6345 <*package>
6346
6347 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6348
6349 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6350 \RequirePackage{l3keys2e}
6351
6352 \keys_define:nn { tikzinput } {
6353   image .bool_set:N = \c_tikzinput_image_bool,
6354   image .default:n = false ,
6355   unknown .code:n = {}
6356 }
6357
6358 \ProcessKeysOptions { tikzinput }
6359
6360 \bool_if:NTF \c_tikzinput_image_bool {
6361   \RequirePackage{graphicx}
6362
6363   \providecommand\usetikzlibrary[]{}
6364   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6365 }{
6366   \RequirePackage{tikz}
6367   \RequirePackage{standalone}
6368
6369   \newcommand \tikzinput [2] [] {
6370     \setkeys{Gin}{#1}
6371     \ifx \Gin@ewidth \Gin@exclamation
6372       \ifx \Gin@eheight \Gin@exclamation
6373         \input { #2 }
6374       \else
6375         \resizebox{!}{ \Gin@eheight }{
6376           \input { #2 }
6377         }
6378       \fi
6379     \else
6380       \ifx \Gin@eheight \Gin@exclamation
6381         \resizebox{ \Gin@ewidth }{!}{
```

```

6382         \input { #2 }
6383     }
6384     \else
6385         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6386             \input { #2 }
6387         }
6388     \fi
6389 \fi
6390 }
6391 }
6392
6393 \newcommand \ctikzinput [2] [] {
6394     \begin{center}
6395         \tikzinput [#1] {#2}
6396     \end{center}
6397 }
6398
6399 \@ifpackageloaded{stex}{
6400     \RequirePackage{stex-tikzinput}
6401 }{}
6402
6403 </package>
6404 <*stex>
6405 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6406 \RequirePackage{stex}
6407 \RequirePackage{tikzinput}
6408
6409 \newcommand\mhtikzinput[2] []{%
6410     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6411     \stex_in_repository:nn\Gin@mhrepos{
6412         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6413     }
6414 }
6415 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6416
6417 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6418     \pgfkeys@spdef\pgf@temp{#1}
6419     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6420     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6421     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6422     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6423     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6424     \catcode'\@=11
6425     \catcode'\|=12
6426     \catcode'\$=3
6427     \pgfutil@InputIfFileExists{#2}{-}{-}
6428     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6429     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6430     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6431 }
6432
6433
6434 \newcommand\libusetikzlibrary[1]{

```

```

6435 \prop_if_exist:NF \l_stex_current_repository_prop {
6436   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6437 }
6438 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6439   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6440 }
6441 \seq_clear:N \l__tikzinput_libinput_files_seq
6442 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6443 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6444
6445 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6446   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6447   \IfFileExists{ \l_tmpa_str }{
6448     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6449   }{}
6450   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6451   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6452 }
6453
6454 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6455 \IfFileExists{ \l_tmpa_str }{
6456   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6457 }{}
6458
6459 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6460   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6461 }{
6462   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6463     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6464       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6465     }
6466   }{
6467     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6468   }
6469 }
6470 }
6471 </stex>

```

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

Chapter 37

document-structure.sty Implementation

```
6472 <*package>
6473 <@@=document_structure>
6474 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6475 \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).

```
6476
6477 \keys_define:nn{ document-structure }{
6478   class      .str_set_x:N = \c_document_structure_class_str,
6479   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6480   unknown    .code:n      = {
6481     \PassOptionsToClass{\CurrentOption}{stex}
6482     \PassOptionsToClass{\CurrentOption}{tikzinput}
6483   }
6484   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6485 }
6486 \ProcessKeysOptions{ document-structure }
6487 \str_if_empty:NT \c_document_structure_class_str {
6488   \str_set:Nn \c_document_structure_class_str {article}
6489 }
6490 \str_if_empty:NT \c_document_structure_topsect_str {
6491   \str_set:Nn \c_document_structure_topsect_str {section}
6492 }
```

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

```
6493 \RequirePackage{xspace}
6494 \RequirePackage{comment}
6495 \RequirePackage{stex}
6496 \AddToHook{begindocument}{
```



```

6497 \ltx@ifpackageloaded{babel}{
6498   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6499   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6500     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6501   }
6502 }{}
6503 }

```

`\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.

```

6504 \int_new:N \l_document_structure_section_level_int
6505 \str_case:NnF \c_document_structure_topsect_str {
6506   {part}}{
6507     \int_set:Nn \l_document_structure_section_level_int {0}
6508   }
6509   {chapter}{
6510     \int_set:Nn \l_document_structure_section_level_int {1}
6511   }
6512 }{
6513   \str_case:NnF \c_document_structure_class_str {
6514     {book}{
6515       \int_set:Nn \l_document_structure_section_level_int {0}
6516     }
6517     {report}{
6518       \int_set:Nn \l_document_structure_section_level_int {0}
6519     }
6520   }{
6521     \int_set:Nn \l_document_structure_section_level_int {2}
6522   }
6523 }

```

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

```

6524 \def\current@section@level{document}%
6525 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6526 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6527 \cs_new_protected:Npn \skipfragment {

```

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

```

6528 \ifcase\l_document_structure_section_level_int
6529 \or\stepcounter{part}
6530 \or\stepcounter{chapter}
6531 \or\stepcounter{section}
6532 \or\stepcounter{subsection}
6533 \or\stepcounter{subsubsection}
6534 \or\stepcounter{paragraph}
6535 \or\stepcounter{subparagraph}
6536 \fi
6537 }

```

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

blindfragment

```

6538 \newcommand\at@begin@blindsfragment[1]{
6539 \newenvironment{blindfragment}
6540 {
6541 \int_incr:N\l_document_structure_section_level_int
6542 \at@begin@blindsfragment\l_document_structure_section_level_int
6543 }{}

```

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

```

6544 \newcommand\sfragment@nonum[2]{
6545 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6546 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6547 }

```

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

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

```

6548 \newcommand\sfragment@num[2]{
6549 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6550 \@nameuse{#1}{#2}
6551 }{
6552 \cs_if_exist:NTF\rdmeta@sectioning{
6553 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6554 }{
6555 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6556 }
6557 }
6558 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6559 }

```

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

sfragment

```

6560 \keys_define:nn { document-structure / sfragment }{
6561 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6562 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6563 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6564 contributors .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6565 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6566 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6567 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6568 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6569 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6570 imports      .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6571 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6572 }
6573 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6574   \str_clear:N \l__document_structure_sfragment_id_str
6575   \str_clear:N \l__document_structure_sfragment_date_str
6576   \clist_clear:N \l__document_structure_sfragment_creators_clist
6577   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6578   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6579   \tl_clear:N \l__document_structure_sfragment_type_tl
6580   \tl_clear:N \l__document_structure_sfragment_short_tl
6581   \tl_clear:N \l__document_structure_sfragment_display_tl
6582   \tl_clear:N \l__document_structure_sfragment_imports_tl
6583   \tl_clear:N \l__document_structure_sfragment_intro_tl
6584   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6585   \keys_set:nn { document-structure / sfragment } { #1 }
6586 }

```

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`

```

6587 \newif@if@mainmatter\@mainmattertrue
6588 \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.

```

6589 \keys_define:nn { document-structure / sectioning }{
6590   name      .str_set_x:N = \l__document_structure_sect_name_str ,
6591   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
6592   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6593   clear     .default:n   = {true} ,
6594   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6595   num       .default:n   = {true}
6596 }
6597 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6598   \str_clear:N \l__document_structure_sect_name_str
6599   \str_clear:N \l__document_structure_sect_ref_str
6600   \bool_set_false:N \l__document_structure_sect_clear_bool
6601   \bool_set_false:N \l__document_structure_sect_num_bool
6602   \keys_set:nn { document-structure / sectioning } { #1 }
6603 }
6604 \newcommand\omdoc@sectioning[3][]{
6605   \l__document_structure_sect_args:n {#1 }
6606   \let\omdoc@sect@name\l__document_structure_sect_name_str
6607   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6608   \if@mainmatter% numbering not overridden by frontmatter, etc.
6609     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6610     \sfragment@num{#2}{#3}
6611   }{
6612     \sfragment@nonum{#2}{#3}
6613   }
6614   \def\current@section@level{\omdoc@sect@name}
6615 \else
6616   \sfragment@nonum{#2}{#3}
6617 \fi
6618 }% 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.

```

6619 \newcommand\sfragment@redefine@addtocontents[1]{%
6620 %\edef\__document_structureimport{#1}%
6621 %\@for\@I:=\__document_structureimport\do{%
6622 %\edef\@path{\csname module@\@I @path\endcsname}%
6623 %\@ifundefined{tf@toc}\relax%
6624 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6625 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6626 %\def\addcontentsline##1##2##3{%
6627 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6628 %\else% hyperref.sty not loaded
6629 %\def\addcontentsline##1##2##3{%
6630 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6631 %\fi
6632 }% hypreref.sty loaded?

```

now the `sfragment` environment itself. This takes care of the table of contents via the helper macro above and then selects the appropriate sectioning command from `article.cls`. It also registers the current level of sfragments in the `\sfragment@level` counter.

```

6633 \newenvironment{sfragment}[2][ ]% keys, title
6634 {
6635   \__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.

```

6636   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6637
6638   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6639     \sfragment@redefine@addtocontents{
6640       %\@ifundefined{module@id}\used@modules%
6641       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6642     }
6643   }

```

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

```

6644
6645   \stex_document_title:n { #2 }
6646
6647   \int_incr:N\l__document_structure_section_level_int
6648   \ifcase\l__document_structure_section_level_int
6649     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6650     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6651 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6652 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6653 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6654 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6655 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6656 \fi
6657 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6658 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6659   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6660 }
6661 }% for customization
6662 {}

```

and finally, we localize the sections

```

6663 \newcommand\omdoc@part@kw{Part}
6664 \newcommand\omdoc@chapter@kw{Chapter}
6665 \newcommand\omdoc@section@kw{Section}
6666 \newcommand\omdoc@subsection@kw{Subsection}
6667 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6668 \newcommand\omdoc@paragraph@kw{paragraph}
6669 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 Front and Backmatter

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

`\printindex`

```

6670 \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).

```

6671 \cs_if_exist:NTF\frontmatter{
6672   \let\__document_structure_orig_frontmatter\frontmatter
6673   \let\frontmatter\relax
6674 }{
6675   \tl_set:Nn\__document_structure_orig_frontmatter{
6676     \clearpage
6677     \@mainmatterfalse
6678     \pagenumbering{roman}
6679   }
6680 }
6681 \cs_if_exist:NTF\backmatter{
6682   \let\__document_structure_orig_backmatter\backmatter
6683   \let\backmatter\relax
6684 }{
6685   \tl_set:Nn\__document_structure_orig_backmatter{
6686     \clearpage
6687     \@mainmatterfalse

```

```

6688     \pagenumbering{roman}
6689   }
6690 }

```

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.

```

6691 \newenvironment{frontmatter}{
6692   \_document_structure_orig_frontmatter
6693 }{
6694   \cs_if_exist:NTF\mainmatter{
6695     \mainmatter
6696   }{
6697     \clearpage
6698     \@mainmattertrue
6699     \pagenumbering{arabic}
6700   }
6701 }

```

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

```

6702 \newenvironment{backmatter}{
6703   \_document_structure_orig_backmatter
6704 }{
6705   \cs_if_exist:NTF\mainmatter{
6706     \mainmatter
6707   }{
6708     \clearpage
6709     \@mainmattertrue
6710     \pagenumbering{arabic}
6711   }
6712 }

```

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

```

6713 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6714 \def \c__document_structure_document_str{document}
6715 \newcommand\afterprematurestop{}
6716 \def\prematurestop@endsfragment{
6717   \unless\ifx\@currenvir\c__document_structure_document_str
6718     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6719     \expandafter\prematurestop@endsfragment
6720   }
6721 }
6722 \providecommand\prematurestop{
6723   \message{Stopping~sTeX~processing~prematurely}
6724   \prematurestop@endsfragment
6725   \afterprematurestop
6726   \end{document}
6727 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
6728 \RequirePackage{etoolbox}
6729 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6730 \newrobustcmd\useSGvar[1]{%
6731   \@ifundefined{sTeX@Gvar@#1}
6732   {\PackageError{document-structure}
6733    {The sTeX Global variable #1 is undefined}
6734    {set it with \protect\setSGvar}}
6735   \@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.

```
6736 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6737   \@ifundefined{sTeX@Gvar@#1}
6738   {\PackageError{document-structure}
6739    {The sTeX Global variable #1 is undefined}
6740    {set it with \protect\setSGvar}}
6741   {\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.

```
6742 \*cls)
6743 \@@=notesslides)
6744 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6745 \RequirePackage{13keys2e}
6746
6747 \keys_define:nn{notesslides / cls}{
6748   class .str_set_x:N = \c__notesslides_class_str,
6749   notes .bool_set:N = \c__notesslides_notes_bool ,
6750   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6751   docopt .str_set_x:N = \c__notesslides_docopt_str,
6752   unknown .code:n = {
6753     \PassOptionsToPackage{\CurrentOption}{document-structure}
6754     \PassOptionsToClass{\CurrentOption}{beamer}
6755     \PassOptionsToPackage{\CurrentOption}{notesslides}
6756     \PassOptionsToPackage{\CurrentOption}{stex}
6757   }
6758 }
6759 \ProcessKeysOptions{ notesslides / cls }
6760
6761 \str_if_empty:NF \c__notesslides_class_str {
6762   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6763 }
6764
6765 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6766   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6767 }
6768 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6769   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6770 }
6771
6772 \RequirePackage{stex}
```



```

6773 \stex_html_backend:T {
6774   \bool_set_true:N\c__notesslides_notes_bool
6775 }
6776
6777 \bool_if:NTF \c__notesslides_notes_bool {
6778   \PassOptionsToPackage{notes=true}{notesslides}
6779 }{
6780   \PassOptionsToPackage{notes=false}{notesslides}
6781 }
6782 \</cls>

```

now we do the same for the notesslides package.

```

6783 \*package>
6784 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6785 \RequirePackage{13keys2e}
6786
6787 \keys_define:nn{notesslides / pkg}{
6788   topsect      .str_set_x:N = \c__notesslides_topsect_str,
6789   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
6790   notes        .bool_set:N = \c__notesslides_notes_bool ,
6791   slides        .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6792   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6793   frameimages .bool_set:N = \c__notesslides_frameimages_bool ,
6794   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6795   nopproblems .bool_set:N = \c__notesslides_nopproblems_bool,
6796   unknown      .code:n      = {
6797     \PassOptionsToClass{\CurrentOption}{stex}
6798     \PassOptionsToClass{\CurrentOption}{tikzinput}
6799   }
6800 }
6801 \ProcessKeysOptions{ notesslides / pkg }
6802
6803 \RequirePackage{stex}
6804 \stex_html_backend:T {
6805   \bool_set_true:N\c__notesslides_notes_bool
6806 }
6807
6808 \newif\ifnotes
6809 \bool_if:NTF \c__notesslides_notes_bool {
6810   \notesttrue
6811 }{
6812   \notesfalse
6813 }
6814

```

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

```

6815 \str_if_empty:NTF \c__notesslides_topsect_str {
6816   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6817 }{
6818   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6819 }
6820 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}
6821 \</package>

```

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

```

6822 <*cls>
6823 \bool_if:NTF \c__notesslides_notes_bool {
6824   \str_if_empty:NT \c__notesslides_class_str {
6825     \str_set:Nn \c__notesslides_class_str {article}
6826   }
6827   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6828     {\c__notesslides_class_str}
6829 }{
6830   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6831   \newcounter{Item}
6832   \newcounter{paragraph}
6833   \newcounter{subparagraph}
6834   \newcounter{Hfootnote}
6835 }
6836 \RequirePackage{document-structure}

```

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

```

6837 \RequirePackage{notesslides}
6838 </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.

```

6839 <*package>
6840 \bool_if:NT \c__notesslides_notes_bool {
6841   \RequirePackage{a4wide}
6842   \RequirePackage{marginnote}
6843   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6844   \RequirePackage{mdframed}
6845   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6846   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6847 }
6848 \RequirePackage{stex-tikzinput}
6849 \RequirePackage{etoolbox}
6850 \RequirePackage{amssymb}
6851 \RequirePackage{amsmath}
6852 \RequirePackage{comment}
6853 \RequirePackage{textcomp}
6854 \RequirePackage{url}
6855 \RequirePackage{graphicx}
6856 \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

```

6857 \bool_if:NT \c__notesslides_notes_bool {
6858   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6859 }
6860
6861
6862 \NewDocumentCommand \libusetheme {0{} m} {
6863   \bool_if:NTF \c__notesslides_notes_bool {
6864     \libusepackage[#1]{beamernotestheme#2}
6865   }{
6866     \libusepackage[#1]{beamertheme#2}
6867   }
6868 }

```

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

```

6869 \newcounter{slide}
6870 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6871 \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.

```

6872 \bool_if:NTF \c__notesslides_notes_bool {
6873   \renewenvironment{note}{\ignorespaces}{}
6874 }{
6875   \excludecomment{note}
6876 }

```

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.

```

6877 \bool_if:NT \c__notesslides_notes_bool {
6878   \newlength{\slideframewidth}
6879   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6880 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6881   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6882     \bool_set_true:N #1
6883   }{
6884     \bool_set_false:N #1
6885   }
6886 }
6887 \keys_define:nn{notesslides / frame}{
6888   label .str_set_x:N = \l__notesslides_frame_label_str,
6889   allowframebreaks .code:n = {
6890     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6891   },
6892   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.

```

6893     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6894 },
6895 fragile .code:n = {
6896     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6897 },
6898 shrink .code:n = {
6899     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6900 },
6901 squeeze .code:n = {
6902     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6903 },
6904 t .code:n = {
6905     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6906 },
6907 }
6908 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6909     \str\_clear:N \l\_notesslides\_frame\_label\_str
6910     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6911     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6912     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6913     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6914     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6915     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6916     \keys\_set:nn { notesslides / frame }{ #1 }
6917 }

```

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

```

6918 \renewenvironment{frame}[1][]{
6919     \_notesslides\_frame\_args:n{#1}
6920     \sffamily
6921     \stepcounter{slide}
6922     \def\@currentlabel{\theslide}
6923     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6924         \label{\l\_notesslides\_frame\_label\_str}
6925     }

```

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

```

6926 \def\itemize@level{outer}
6927 \def\itemize@outer{outer}
6928 \def\itemize@inner{inner}
6929 \renewcommand\newpage{\addtocounter{framenum}{1}}
6930 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6931 \renewenvironment{itemize}{
6932     \ifx\itemize@level\itemize@outer
6933         \def\itemize@label{\$ \rhd \$}
6934     \fi
6935     \ifx\itemize@level\itemize@inner
6936         \def\itemize@label{\$ \scriptstyle \rhd \$}
6937     \fi
6938     \begin{list}
6939     {\itemize@label}
6940     {\setlength{\labelsep}{.3em}
6941      \setlength{\labelwidth}{.5em}
6942      \setlength{\leftmargin}{1.5em}
6943     }

```

```

6944 \edef\itemize@level{\itemize@inner}
6945 }{
6946 \end{list}
6947 }

```

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

```

6948 \stex_html_backend:TF {
6949 \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6950 }{
6951 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6952 }
6953 }{
6954 \stex_html_backend:TF {
6955 \miko@slidelabel\egroup\end{stex_annotate_env}
6956 }\medskip\miko@slidelabel\end{mdframed}}
6957 }

```

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

`\frametitle`

```

6958 \renewcommand{\frametitle}[1]{
6959 \stex_document_title:n { #1 }
6960 {\Large\bf\sf\color{blue}{#1}}\medskip
6961 }
6962 }

```

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

EdN:19

`\pause`

19

```

6963 \bool_if:NT \c__notesslides_notes_bool {
6964 \newcommand\pause{}
6965 }

```

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

`nparagraph`

```

6966 \bool_if:NTF \c__notesslides_notes_bool {
6967 \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]]{\end{sparagraph}}
6968 }{
6969 \excludecomment{nparagraph}
6970 }

```

`nfragment`

```

6971 \bool_if:NTF \c__notesslides_notes_bool {
6972 \newenvironment{nfragment}[2][\begin{sfragment}[#1]{#2}]{\end{sfragment}}
6973 }{
6974 \excludecomment{nfragment}
6975 }

```

`ndefinition`

```

6976 \bool_if:NTF \c__notesslides_notes_bool {
6977 \newenvironment{ndefinition}[1][\begin{sdefinition}[#1]]{\end{sdefinition}}
6978 }{
6979 \excludecomment{ndefinition}
6980 }

```

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

nassertion

```
6981 \bool_if:NTF \c_notesslides_notes_bool {
6982   \newenvironment{nassertion}[1][\begin{sassertion}[#1]]{\end{sassertion}}
6983 }{
6984   \excludecomment{nassertion}
6985 }
```

nsproof

```
6986 \bool_if:NTF \c_notesslides_notes_bool {
6987   \newenvironment{nproof}[2][\begin{sproof}[#1][#2]]{\end{sproof}}
6988 }{
6989   \excludecomment{nproof}
6990 }
```

nexample

```
6991 \bool_if:NTF \c_notesslides_notes_bool {
6992   \newenvironment{nexample}[1][\begin{sexample}[#1]]{\end{sexample}}
6993 }{
6994   \excludecomment{nexample}
6995 }
```

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

```
6996 \def\inputref@preskip{\smallskip}
6997 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
6998 \let\orig@inputref\inputref
6999 \def\inputref{\@ifstar\ninputref\orig@inputref}
7000 \newcommand\ninputref[2][\{
7001   \bool_if:NT \c_notesslides_notes_bool {
7002     \orig@inputref[#1][#2]
7003   }
7004 }
```

(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.

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

```
7005 \newlength{\slidelogoheight}
7006
7007 \bool_if:NTF \c_notesslides_notes_bool {
7008   \setlength{\slidelogoheight}{.4cm}
7009 }{
7010   \setlength{\slidelogoheight}{1cm}
7011 }
7012 \newsavebox{\slidelogo}
```

```

7013 \sbox{\slidologo}{\TeX}
7014 \newrobustcmd{\setslidologo}[1]{
7015   \sbox{\slidologo}{\includegraphics[height=\slidologoheight]{#1}}
7016 }

```

(End definition for `\setslidologo`. 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.

```

7017 \def\source{Michael Kohlhase}% customize locally
7018 \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.

```

7019 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7020 \newsavebox{\cclogo}
7021 \sbox{\cclogo}{\includegraphics[height=\slidologoheight]{stex-cc_somerights}}
7022 \newif\ifcchref\cchreffalse
7023 \AtBeginDocument{
7024   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7025 }
7026 \def\licensing{
7027   \ifcchref
7028     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7029   \else
7030     {\usebox{\cclogo}}
7031   \fi
7032 }
7033 \newrobustcmd{\setlicensing}[2][{}]{
7034   \def\@url{#1}
7035   \sbox{\cclogo}{\includegraphics[height=\slidologoheight]{#2}}
7036   \ifx\@url\@empty
7037     \def\licensing{{\usebox{\cclogo}}}
7038   \else
7039     \def\licensing{
7040       \ifcchref
7041         \href{#1}{\usebox{\cclogo}}
7042       \else
7043         {\usebox{\cclogo}}
7044       \fi
7045     }
7046   \fi
7047 }

```

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

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

```

7048 \newrobustcmd\miko@slidelabel{
7049   \vbox to \slidologoheight{

```

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

```

7050 \vss\hbox to \slidewidth
7051 {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slide\logo}}
7052 }
7053 }

```

(End definition for \slide\label. 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.

```

7054 \def\Gin@mhrepos{}
7055 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7056 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7057 \newrobustcmd\frameimage[2][{}]{
7058   \stepcounter{slide}
7059   \bool_if:NT \c__notesslides_frameimages_bool {
7060     \def\Gin@ewidth{} \setkeys{Gin}{#1}
7061     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7062     \begin{center}
7063       \bool_if:NTF \c__notesslides_fiboxed_bool {
7064         \fbox{
7065           \ifx\Gin@ewidth\@empty
7066             \ifx\Gin@mhrepos\@empty
7067               \mhgraphics[width=\slidewidth,#1]{#2}
7068             \else
7069               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7070             \fi
7071           \else% Gin@ewidth empty
7072             \ifx\Gin@mhrepos\@empty
7073               \mhgraphics[#1]{#2}
7074             \else
7075               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7076             \fi
7077           \fi% Gin@ewidth empty
7078         }
7079       }{
7080         \ifx\Gin@ewidth\@empty
7081           \ifx\Gin@mhrepos\@empty
7082             \mhgraphics[width=\slidewidth,#1]{#2}
7083           \else
7084             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7085           \fi
7086         \ifx\Gin@mhrepos\@empty
7087           \mhgraphics[#1]{#2}
7088         \else
7089           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7090         \fi
7091         \fi% Gin@ewidth empty
7092       }
7093     \end{center}
7094     \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7095     \bool_if:NF \c__notesslides_notes_bool { \vfill }

```



```

7096 }
7097 } % 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.

```

7098 \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

```

7099 \AddToHook{begindocument}{
7100   \definecolor{green}{rgb}{0,.5,0}
7101   \definecolor{purple}{cmk}{.3,1,0,.17}
7102 }

```

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`.

```

7103 % \def\STpresent#1{\textcolor{blue}{#1}}
7104 \def\defemph#1{\textcolor{magenta}{#1}}
7105 \def\symrefemph#1{\textcolor{cyan}{#1}}
7106 \def\compemph#1{\textcolor{blue}{#1}}
7107 \def\titleemph#1{\textcolor{blue}{#1}}
7108 \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.

```

7109 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7110 \def\smalltextwarning{
7111   \pgfuseimage{miko@small@dbend}
7112   \xspace
7113 }
7114 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7115 \newrobustcmd\textwarning{
7116   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7117   \xspace
7118 }
7119 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7120 \newrobustcmd\bigtextwarning{
7121   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7122   \xspace
7123 }

```

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

```

7124 \newrobustcmd\putgraphicsat[3]{
7125   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7126 }
7127 \newrobustcmd\putat[2]{
7128   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7129 }

```

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`.

```

7130 \bool_if:NT \c__notesslides_sectocframes_bool {
7131   \str_if_eq:VnTF \__notesslidestopsect{part}{
7132     \newcounter{chapter}\counterwithin*{section}{chapter}
7133   }{
7134     \str_if_eq:VnTF \__notesslidestopsect{chapter}{
7135       \newcounter{chapter}\counterwithin*{section}{chapter}
7136     }
7137   }
7138 }

```

`\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
7139 \def\part@prefix{}
7140 \@ifpackageloaded{document-structure}{}{
7141   \str_case:VnF \__notesslidestopsect {
7142     {part}{
7143       \int_set:Nn \l_document_structure_section_level_int {0}
7144       \def\thesection{\arabic{chapter}.\arabic{section}}
7145       \def\part@prefix{\arabic{chapter}.}
7146     }
7147     {chapter}{
7148       \int_set:Nn \l_document_structure_section_level_int {1}
7149       \def\thesection{\arabic{chapter}.\arabic{section}}
7150       \def\part@prefix{\arabic{chapter}.}
7151     }
7152   }{
7153     \int_set:Nn \l_document_structure_section_level_int {2}
7154     \def\part@prefix{}
7155   }
7156 }
7157
7158 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `sfragment` environment that chooses the \LaTeX sectioning macros according to `\section@level`.

sfragment

```

7159 \renewenvironment{sfragment}[2][]{
7160   \__document_structure_sfragment_args:n { #1 }
7161   \int_incr:N \l_document_structure_section_level_int
7162   \bool_if:NT \c__notesslides_sectocframes_bool {
7163     \stepcounter{slide}
7164     \begin{frame}[noframenumbering]
7165     \vfill\Large\centering
7166     \red{
7167       \ifcase\l_document_structure_section_level_int\or

```

```

7168         \stepcounter{part}
7169         \def\__notesslideslabel{\o{omdoc@part@kw}~\Roman{part}}
7170         \def\currentsectionlevel{\o{omdoc@part@kw}}
7171     \or
7172         \stepcounter{chapter}
7173         \def\__notesslideslabel{\o{omdoc@chapter@kw}~\arabic{chapter}}
7174         \def\currentsectionlevel{\o{omdoc@chapter@kw}}
7175     \or
7176         \stepcounter{section}
7177         \def\__notesslideslabel{\part@prefix\arabic{section}}
7178         \def\currentsectionlevel{\o{omdoc@section@kw}}
7179     \or
7180         \stepcounter{subsection}
7181         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7182         \def\currentsectionlevel{\o{omdoc@subsection@kw}}
7183     \or
7184         \stepcounter{subsubsection}
7185         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7186         \def\currentsectionlevel{\o{omdoc@subsubsection@kw}}
7187     \or
7188         \stepcounter{paragraph}
7189         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7190         \def\currentsectionlevel{\o{omdoc@paragraph@kw}}
7191     \else
7192         \def\__notesslideslabel{}
7193         \def\currentsectionlevel{\o{omdoc@paragraph@kw}}
7194     \fi% end ifcase
7195     \__notesslideslabel%\sref@label@id\__notesslideslabel
7196     \quad #2%
7197 }%
7198 \vfill%
7199 \end{frame}%
7200 }
7201 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7202     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7203 }
7204 }{}
7205 }

```

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

```

7206 \def\inserttheorembodyfont{\normalfont}
7207 %\bool_if:NF \c__notesslides_notes_bool {
7208 % \defbeamertemplate{theorem begin}{miko}
7209 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7210 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7211 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7212 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7213 % \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.

```

7214 % \expandafter\def\csname Parent2\endcsname{}

```

```

7215 %}
7216
7217 \AddToHook{begindocument}{% this does not work for some reasons
7218   \setbeamertemplate{theorems}[ams style]
7219 }
7220 \bool_if:NT \c__notesslides_notes_bool {
7221   \renewenvironment{columns}[1][]{%
7222     \par\noindent%
7223     \begin{minipage}%
7224       \slidewidth\centering\leavevmode%
7225   }{%
7226     \end{minipage}\par\noindent%
7227   }%
7228   \newsavebox\columnbox%
7229   \renewenvironment<>{column}[2][]{%
7230     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7231   }{%
7232     \end{minipage}\end{lrbox}\usebox\columnbox%
7233   }%
7234 }
7235 \bool_if:NTF \c__notesslides_noproblems_bool {
7236   \newenvironment{problems}{}{}
7237 }{
7238   \excludcomment{problems}
7239 }

```

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.

```

7240 \gdef\printexcursions{}
7241 \newcommand\excursionref[2]{% label, text
7242   \bool_if:NT \c__notesslides_notes_bool {
7243     \begin{sparagraph}[title=Excursion]
7244       #2 \sref[fallback=the appendix]{#1}.
7245     \end{sparagraph}
7246   }
7247 }
7248 \newcommand\activate@excursion[2][{}{
7249   \gappto\printexcursions{\inputref[#1]{#2}}
7250 }
7251 \newcommand\excursion[4][{}{repos, label, path, text
7252   \bool_if:NT \c__notesslides_notes_bool {
7253     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7254   }
7255 }

```

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

\excursiongroup

```

7256 \keys_define:nn{notesslides / excursiongroup }{

```

```

7257 id      .str_set_x:N = \l__notesslides_excursion_id_str,
7258 intro   .tl_set:N    = \l__notesslides_excursion_intro_tl,
7259 mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7260 }
7261 \cs_new_protected:Nn \l__notesslides_excursion_args:n {
7262   \tl_clear:N \l__notesslides_excursion_intro_tl
7263   \str_clear:N \l__notesslides_excursion_id_str
7264   \str_clear:N \l__notesslides_excursion_mhrepos_str
7265   \keys_set:nn {notesslides / excursiongroup} { #1 }
7266 }
7267 \newcommand\excursiongroup[1][ ]{
7268   \l__notesslides_excursion_args:n{ #1 }
7269   \ifdefempty\printexcursions{}% only if there are excursions
7270   {\begin{note}
7271     \begin{sfragment}[#1]{Excursions}%
7272     \ifdefempty\l__notesslides_excursion_intro_tl}{
7273       \inputref[\l__notesslides_excursion_mhrepos_str]{
7274         \l__notesslides_excursion_intro_tl
7275       }
7276     }
7277     \printexcursions%
7278   \end{sfragment}
7279   \end{note}}
7280 }
7281 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7282 \end{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.

```
7283 <*package>
7284 <@@=problems>
7285 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7286 \RequirePackage{13keys2e,stex}
7287
7288 \keys_define:nn { problem / pkg }{
7289   notes      .default:n    = { true },
7290   notes      .bool_set:N   = \c__problems_notes_bool,
7291   gnotes     .default:n    = { true },
7292   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7293   hints      .default:n    = { true },
7294   hints      .bool_set:N   = \c__problems_hints_bool,
7295   solutions  .default:n    = { true },
7296   solutions  .bool_set:N   = \c__problems_solutions_bool,
7297   pts        .default:n    = { true },
7298   pts        .bool_set:N   = \c__problems_pts_bool,
7299   min        .default:n    = { true },
7300   min        .bool_set:N   = \c__problems_min_bool,
7301   boxed      .default:n    = { true },
7302   boxed      .bool_set:N   = \c__problems_boxed_bool,
7303   unknown    .code:n       = {}
7304 }
7305 \newif\ifsolutions
7306
7307 \ProcessKeysOptions{ problem / pkg }
7308 \bool_if:NTF \c__problems_solutions_bool {
7309   \solutionstrue
7310 }{
7311   \solutionsfalse
7312 }
```

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

```
7313 \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.

```
7314 \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.

```
7315 \def\prob@problem@kw{Problem}
7316 \def\prob@solution@kw{Solution}
7317 \def\prob@hint@kw{Hint}
7318 \def\prob@note@kw{Note}
7319 \def\prob@gnote@kw{Grading}
7320 \def\prob@pt@kw{pt}
7321 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7322 \AddToHook{begindocument}{
7323   \ltx@ifpackageloaded{babel}{
7324     \makeatletter
7325     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7326     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7327       \input{problem-ngerman.ldf}
7328     }
7329     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7330       \input{problem-finnish.ldf}
7331     }
7332     \clist_if_in:NnT \l_tmpa_clist {french}{
7333       \input{problem-french.ldf}
7334     }
7335     \clist_if_in:NnT \l_tmpa_clist {russian}{
7336       \input{problem-russian.ldf}
7337     }
7338     \makeatother
7339   }{}
7340 }
```

39.2 Problems and Solutions

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

```
7341 \keys_define:nn{ problem / problem }{
7342   id      .str_set_x:N = \l__problems_prob_id_str,
7343   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7344   min     .tl_set:N    = \l__problems_prob_min_tl,
7345   title   .tl_set:N    = \l__problems_prob_title_tl,
7346   type    .tl_set:N    = \l__problems_prob_type_tl,
7347   imports .tl_set:N    = \l__problems_prob_imports_tl,
7348   name    .str_set_x:N = \l__problems_prob_name_str,
7349   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7350 }
7351 \cs_new_protected:Nn \__problems_prob_args:n {
7352   \str_clear:N \l__problems_prob_id_str
7353   \str_clear:N \l__problems_prob_name_str
7354   \tl_clear:N \l__problems_prob_pts_tl
7355   \tl_clear:N \l__problems_prob_min_tl
7356   \tl_clear:N \l__problems_prob_title_tl
7357   \tl_clear:N \l__problems_prob_type_tl
7358   \tl_clear:N \l__problems_prob_imports_tl
7359   \int_zero_new:N \l__problems_prob_refnum_int
7360   \keys_set:nn { problem / problem }{ #1 }
7361   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7362     \let\l__problems_prob_refnum_int\undefined
7363   }
7364 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7365 \newcounter{problem}[section]
7366 \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.

```

7367 \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

```

7368 \newcommand\prob@number{
7369   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7370     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7371   }{
7372     \int_if_exist:NTF \l__problems_prob_refnum_int {
7373       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7374     }{
7375       \prob@label\theproblem
7376     }
7377   }
7378 }

```

(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.

```

7379 \newcommand\prob@title[3]{%
7380   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7381     #2 \l__problems_inclprob_title_tl #3
7382   }{
7383     \tl_if_exist:NTF \l__problems_prob_title_tl {
7384       #2 \l__problems_prob_title_tl #3
7385     }{
7386       #1

```



```

7387     }
7388   }
7389 }

```

(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.

```

7390 \def\prob@heading{
7391   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7392   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7393 }

```

(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`

```

7394 \newenvironment{sproblem}[1][{}]{
7395   \__problems_prob_args:n{#1}%\sref@target%
7396   \@in@omtexttrue% we are in a statement (for inline definitions)
7397   \stepcounter{problem}\record@problem
7398   \def\current@section@level{\prob@problem@kw}
7399
7400   \str_if_empty:NT \l__problems_prob_name_str {
7401     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7402     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7403     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7404   }
7405   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7406
7407   \stex_reactivate_macro:N \STEXexport
7408   \stex_reactivate_macro:N \importmodule
7409   \stex_reactivate_macro:N \symdecl
7410   \stex_reactivate_macro:N \notation
7411   \stex_reactivate_macro:N \symdef
7412
7413   \stex_if_do_html:T{
7414     \begin{stex_annotate_env} {problem} {
7415       \l_stex_module_ns_str ? \l_stex_module_name_str
7416     }
7417
7418     \stex_annotate_invisible:nnn{header}{~}{~} {
7419       \stex_annotate:nnn{language}{~}{~} \l_stex_module_lang_str }{~}
7420       \stex_annotate:nnn{signature}{~}{~} \l_stex_module_sig_str }{~}
7421       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7422         \stex_annotate:nnn{metatheory}{~}{~} \l_stex_module_meta_str }{~}
7423     }
7424   }
7425 }
7426

```

```

7427 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7428
7429
7430 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7431   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7432 }{
7433   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7434 }
7435 \str_if_exist:NTF \l__problems_inclprob_id_str {
7436   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7437 }{
7438   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7439 }
7440
7441
7442 \stex_if_smsmode:F {
7443   \clist_set:No \l_tmpa_clist \sproblemtype
7444   \tl_clear:N \l_tmpa_tl
7445   \clist_map_inline:Nn \l_tmpa_clist {
7446     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7447       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7448     }
7449   }
7450   \tl_if_empty:NTF \l_tmpa_tl {
7451     \__problems_sproblem_start:
7452   }{
7453     \l_tmpa_tl
7454   }
7455 }
7456 \stex_ref_new_doc_target:n \sproblemid
7457 \stex_smsmode_do:
7458 }{
7459   \__stex_modules_end_module:
7460   \stex_if_smsmode:F{
7461     \clist_set:No \l_tmpa_clist \sproblemtype
7462     \tl_clear:N \l_tmpa_tl
7463     \clist_map_inline:Nn \l_tmpa_clist {
7464       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7465         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7466       }
7467     }
7468     \tl_if_empty:NTF \l_tmpa_tl {
7469       \__problems_sproblem_end:
7470     }{
7471       \l_tmpa_tl
7472     }
7473   }
7474   \stex_if_do_html:T{
7475     \end{stex_annotate_env}
7476   }
7477
7478 \smallskip
7479 }
7480

```

```

7481 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7482
7483
7484
7485 \cs_new_protected:Nn \__problems_sproblem_start: {
7486   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7487 }
7488 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7489
7490 \newcommand\stexpatchproblem[3][] {
7491   \str_set:Nx \l_tmpa_str{ #1 }
7492   \str_if_empty:NTF \l_tmpa_str {
7493     \tl_set:Nn \__problems_sproblem_start: { #2 }
7494     \tl_set:Nn \__problems_sproblem_end: { #3 }
7495   }{
7496     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7497     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7498   }
7499 }
7500
7501
7502 \bool_if:NT \c__problems_boxed_bool {
7503   \surroundwithmdframed{problem}
7504 }

```

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

```

7505 \def\record@problem{
7506   \protected@write\@auxout{}
7507   {
7508     \string\@problem{\prob@number}
7509     {
7510       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7511         \l__problems_inclprob_pts_tl
7512       }{
7513         \l__problems_prob_pts_tl
7514       }
7515     }%
7516     {
7517       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7518         \l__problems_inclprob_min_tl
7519       }{
7520         \l__problems_prob_min_tl
7521       }
7522     }
7523   }
7524 }

```

(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).

```

7525 \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 its own keys that we need to define first.

```

7526 \keys_define:nn { problem / solution }{
7527   id          .str_set_x:N = \l__problems_solution_id_str ,
7528   for         .tl_set:N    = \l__problems_solution_for_tl ,
7529   height      .dim_set:N   = \l__problems_solution_height_dim ,
7530   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7531   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7532   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7533 }
7534 \cs_new_protected:Nn \__problems_solution_args:n {
7535   \str_clear:N \l__problems_solution_id_str
7536   \tl_clear:N \l__problems_solution_for_tl
7537   \tl_clear:N \l__problems_solution_srccite_tl
7538   \clist_clear:N \l__problems_solution_creators_clist
7539   \clist_clear:N \l__problems_solution_contributors_clist
7540   \dim_zero:N \l__problems_solution_height_dim
7541   \keys_set:nn { problem / solution }{ #1 }
7542 }

```

the next step is to define a helper macro that does what is needed to start a solution.

```

7543 \newcommand\@startsolution[1][ ]{
7544   \__problems_solution_args:n { #1 }
7545   \@in@omtexttrue% we are in a statement.
7546   \bool_if:NF \c__problems_boxed_bool { \hrule }
7547   \smallskip\noindent
7548   {\textbf\prob@solution@kw : \enspace}
7549   \begin{small}
7550   \def\current@section@level{\prob@solution@kw}
7551   \ignorespacesandpars
7552 }

```

\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.

```

7553 \box_new:N \l__problems_solution_box
7554 \newenvironment{solution}[1][ ]{
7555   \stex_html_backend:TF{
7556     \stex_if_do_html:T{
7557       \begin{stex_annotate_env}{solution}{ }
7558     }
7559   }{
7560     \setbox\l__problems_solution_box\vbox\bgroup
7561     \par\smallskip\hrule\smallskip
7562     \noindent\textbf{Solution:}~
7563   }
7564 }{
7565   \stex_html_backend:TF{
7566     \stex_if_do_html:T{
7567       \end{stex_annotate_env}
7568     }
7569   }{

```

```

7570 \smallskip\hrule
7571 \egroup
7572 \bool_if:NT \c__problems_solutions_bool {
7573   \box\l__problems_solution_box
7574 }
7575 }
7576 }
7577
7578 \newcommand\startsolutions{
7579   \bool_set_true:N \c__problems_solutions_bool
7580   % \specialcomment{solution}{\@startsolution}{
7581   %   \bool_if:NF \c__problems_boxed_bool {
7582   %     \hrule\medskip
7583   %   }
7584   %   \end{small}%
7585   % }
7586   % \bool_if:NT \c__problems_boxed_bool {
7587   %   \surroundwithmdframed{solution}
7588   % }
7589 }

```

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

\stopsolutions

```

7590 \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.

```

7591 \ifsolutions
7592   \startsolutions
7593 \else
7594   \stopsolutions
7595 \fi

```

exnote

```

7596 \bool_if:NFT \c__problems_notes_bool {
7597   \newenvironment{exnote}[1][]{
7598     \par\smallskip\hrule\smallskip
7599     \noindent\textbf{\prob@note@kw :~ }\small
7600   }{
7601     \smallskip\hrule
7602   }
7603 }{
7604   \excludecomment{exnote}
7605 }

```

hint

```

7606 \bool_if:NFT \c__problems_notes_bool {
7607   \newenvironment{hint}[1][]{
7608     \par\smallskip\hrule\smallskip
7609     \noindent\textbf{\prob@hint@kw :~ }\small
7610   }{
7611     \smallskip\hrule
7612   }

```

```

7613 \newenvironment{exhint}[1][]{
7614   \par\smallskip\hrule\smallskip
7615   \noindent\textbf{\prob@hint@kw :~ }\small
7616 }{
7617   \smallskip\hrule
7618 }
7619 }{
7620   \excludecomment{hint}
7621   \excludecomment{exhint}
7622 }

```

gnote

```

7623 \bool_if:NTF \c__problems_notes_bool {
7624   \newenvironment{gnote}[1][]{
7625     \par\smallskip\hrule\smallskip
7626     \noindent\textbf{\prob@gnote@kw :~ }\small
7627   }{
7628     \smallskip\hrule
7629   }
7630   }{
7631     \excludecomment{gnote}
7632   }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7633 \newenvironment{mcb}{
7634   \begin{enumerate}
7635 }{
7636   \end{enumerate}
7637 }

```

we define the keys for the mcb macro

```

7638 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7639   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7640     \bool_set_true:N #1
7641   }{
7642     \bool_set_false:N #1
7643   }
7644 }
7645 \keys_define:nn { problem / mcb }{
7646   id .str_set:x:N = \l__problems_mcc_id_str ,
7647   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7648   T .default:n = { false } ,
7649   T .bool_set:N = \l__problems_mcc_t_bool ,
7650   F .default:n = { false } ,
7651   F .bool_set:N = \l__problems_mcc_f_bool ,
7652   Ttext .tl_set:N = \l__problems_mcc_Ttext_str ,
7653   Ftext .tl_set:N = \l__problems_mcc_Ftext_str
7654 }
7655 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

²¹EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7656 \str_clear:N \l__problems_mcc_id_str
7657 \tl_clear:N \l__problems_mcc_feedback_tl
7658 \bool_set_false:N \l__problems_mcc_t_bool
7659 \bool_set_false:N \l__problems_mcc_f_bool
7660 \tl_clear:N \l__problems_mcc_Ttext_tl
7661 \tl_clear:N \l__problems_mcc_Ftext_tl
7662 \str_clear:N \l__problems_mcc_id_str
7663 \keys_set:nn { problem / mcc }{ #1 }
7664 }

```

\mcc

```

7665 \def\mccTrueText{\textbf{(true)~}}
7666 \def\mccFalseText{\textbf{(false)~}}
7667 \newcommand\mcc[2][] {
7668   \l__problems_mcc_args:n{ #1 }
7669   \item[{$\Box$}] #2
7670   \ifsolutions
7671     \l
7672     \bool_if:NT \l__problems_mcc_t_bool {
7673       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7674     }
7675     \bool_if:NT \l__problems_mcc_f_bool {
7676       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7677     }
7678     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7679       \emph{(\l__problems_mcc_feedback_tl)}
7680     }
7681   \fi
7682 } %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.

```

7683
7684 \keys_define:nn{ problem / inclproblem }{
7685   id      .str_set_x:N = \l__problems_inclprob_id_str,
7686   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7687   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7688   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7689   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7690   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7691   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7692 }
7693 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7694   \str_clear:N \l__problems_prob_id_str
7695   \tl_clear:N \l__problems_inclprob_pts_tl
7696   \tl_clear:N \l__problems_inclprob_min_tl
7697   \tl_clear:N \l__problems_inclprob_title_tl
7698   \tl_clear:N \l__problems_inclprob_type_tl

```

```

7699 \int_zero_new:N \l__problems_inclprob_refnum_int
7700 \str_clear:N \l__problems_inclprob_mhrepos_str
7701 \keys_set:nn { problem / inclproblem }{ #1 }
7702 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7703   \let\l__problems_inclprob_pts_tl\undefined
7704 }
7705 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7706   \let\l__problems_inclprob_min_tl\undefined
7707 }
7708 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7709   \let\l__problems_inclprob_title_tl\undefined
7710 }
7711 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7712   \let\l__problems_inclprob_type_tl\undefined
7713 }
7714 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7715   \let\l__problems_inclprob_refnum_int\undefined
7716 }
7717 }
7718
7719 \cs_new_protected:Nn \__problems_inclprob_clear: {
7720   \let\l__problems_inclprob_id_str\undefined
7721   \let\l__problems_inclprob_pts_tl\undefined
7722   \let\l__problems_inclprob_min_tl\undefined
7723   \let\l__problems_inclprob_title_tl\undefined
7724   \let\l__problems_inclprob_type_tl\undefined
7725   \let\l__problems_inclprob_refnum_int\undefined
7726   \let\l__problems_inclprob_mhrepos_str\undefined
7727 }
7728 \__problems_inclprob_clear:
7729
7730 \newcommand\includeproblem[2][ ]{
7731   \__problems_inclprob_args:n{ #1 }
7732   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7733     \stex_html_backend:TF {
7734       \str_clear:N \l_tmpa_str
7735       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7736         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7737       }
7738       \stex_annotate_invisible:nnn{includeproblem}{
7739         \l_tmpa_str / #2
7740       }{}
7741     }{
7742       \begingroup
7743       \inputreftrue
7744       \tl_if_empty:nTF{ ##1 }{
7745         \input{#2}
7746       }{
7747         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7748       }
7749       \endgroup
7750     }
7751   }
7752   \__problems_inclprob_clear:

```



```
7753 }
```

(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.

```
7754 \AddToHook{enddocument}{
7755   \bool_if:NT \c__problems_pts_bool {
7756     \message{Total:~\arabic{pts}~points}
7757   }
7758   \bool_if:NT \c__problems_min_bool {
7759     \message{Total:~\arabic{min}~minutes}
7760   }
7761 }
```

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

```
7762 \def\pts#1{
7763   \bool_if:NT \c__problems_pts_bool {
7764     \marginpar{#1~\prob@pt@kw}
7765   }
7766 }
7767 \def\min#1{
7768   \bool_if:NT \c__problems_min_bool {
7769     \marginpar{#1~\prob@min@kw}
7770   }
7771 }
```

`\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.

```
7772 \newcounter{pts}
7773 \def\show@pts{
7774   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7775     \bool_if:NT \c__problems_pts_bool {
7776       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7777       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7778     }
7779   }{
7780     \tl_if_exist:NT \l__problems_prob_pts_tl {
7781       \bool_if:NT \c__problems_pts_bool {
7782         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7783           \tl_set:Nn \l__problems_prob_pts_tl {0}
7784         }
7785         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7786         \addtocounter{pts}{\l__problems_prob_pts_tl}
7787       }
7788     }
7789   }
7790 }
```

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

\show@min

```

7791 \newcounter{min}
7792 \def\show@min{
7793   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7794     \bool_if:NT \c__problems_min_bool {
7795       \marginpar{\l__problems_inclprob_pts_tl\ min}
7796       \addtocounter{min}{\l__problems_inclprob_min_tl}
7797     }
7798   }{
7799     \tl_if_exist:NT \l__problems_prob_min_tl {
7800       \bool_if:NT \c__problems_min_bool {
7801         \tl_if_empty:NT\l__problems_prob_min_tl{
7802           \tl_set:Nn \l__problems_prob_min_tl {0}
7803         }
7804         \marginpar{\l__problems_prob_min_tl\ min}
7805         \addtocounter{min}{\l__problems_prob_min_tl}
7806       }
7807     }
7808   }
7809 }
7810 \end{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.

```
7811 \*package>
7812 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7813 \RequirePackage{13keys2e}
7814
7815 \newif\iftest\testfalse
7816 \DeclareOption{test}{\testtrue}
7817 \newif\ifmultiple\multiplefalse
7818 \DeclareOption{multiple}{\multipletrue}
7819 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7820 \ProcessOptions
```

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

```
7821 \RequirePackage{keyval}[1997/11/10]
7822 \RequirePackage{problem}
```

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

```
7823 \newcommand\hwexam@assignment@kw{Assignment}
7824 \newcommand\hwexam@given@kw{Given}
7825 \newcommand\hwexam@due@kw{Due}
7826 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7827 blank~for~extra~space}
7828 \def\hwexam@minutes@kw{minutes}
7829 \newcommand\correction@probs@kw{prob.}
7830 \newcommand\correction@pts@kw{total}
7831 \newcommand\correction@reached@kw{reached}
7832 \newcommand\correction@sum@kw{Sum}
7833 \newcommand\correction@grade@kw{grade}
7834 \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

```

7835 \AddToHook{begindocument}{
7836 \ltx@ifpackageloaded{babel}{
7837 \makeatletter
7838 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7839 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7840 \input{hwexam-ngerman.ldf}
7841 }
7842 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7843 \input{hwexam-finnish.ldf}
7844 }
7845 \clist_if_in:NnT \l_tmpa_clist {french}{
7846 \input{hwexam-french.ldf}
7847 }
7848 \clist_if_in:NnT \l_tmpa_clist {russian}{
7849 \input{hwexam-russian.ldf}
7850 }
7851 \makeatother
7852 }{}
7853 }
7854

```

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.

```

7855 \newcounter{assignment}
7856 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7857 \keys_define:nn { hwexam / assignment } {
7858 id .str_set:N = \l_@@_assign_id_str,
7859 number .int_set:N = \l_@@_assign_number_int,
7860 title .tl_set:N = \l_@@_assign_title_tl,
7861 type .tl_set:N = \l_@@_assign_type_tl,
7862 given .tl_set:N = \l_@@_assign_given_tl,
7863 due .tl_set:N = \l_@@_assign_due_tl,
7864 loadmodules .code:n = {
7865 \bool_set_true:N \l_@@_assign_loadmodules_bool
7866 }
7867 }
7868 \cs_new_protected:Nn \_@@_assignment_args:n {
7869 \str_clear:N \l_@@_assign_id_str
7870 \int_set:Nn \l_@@_assign_number_int {-1}
7871 \tl_clear:N \l_@@_assign_title_tl
7872 \tl_clear:N \l_@@_assign_type_tl
7873 \tl_clear:N \l_@@_assign_given_tl
7874 \tl_clear:N \l_@@_assign_due_tl
7875 \bool_set_false:N \l_@@_assign_loadmodules_bool
7876 \keys_set:nn { hwexam / assignment }{ #1 }
7877 }

```

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.

```

7878 \newcommand\given@due[2]{
7879 \bool_lazy_all:nF {
7880 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7881 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7882 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7883 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7884 }{ #1 }
7885
7886 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7887 \tl_if_empty:NF \l_@@_assign_given_tl {
7888 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7889 }
7890 }{
7891 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7892 }
7893
7894 \bool_lazy_or:nnF {
7895 \bool_lazy_and_p:nn {
7896 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7897 }{
7898 \tl_if_empty_p:V \l_@@_assign_due_tl
7899 }
7900 }{
7901 \bool_lazy_and_p:nn {
7902 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7903 }{
7904 \tl_if_empty_p:V \l_@@_assign_due_tl
7905 }
7906 }{ ,~ }
7907
7908 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7909 \tl_if_empty:NF \l_@@_assign_due_tl {
7910 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7911 }
7912 }{
7913 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7914 }
7915
7916 \bool_lazy_all:nF {
7917 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7918 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7919 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7920 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7921 }{ #2 }
7922 }

```

`\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.

```

7923 \newcommand\assignment@title[3]{
7924 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7925 \tl_if_empty:NTF \l_@@_assign_title_tl {
7926 #1
7927 }{
7928 #2\l_@@_assign_title_tl#3
7929 }
7930 }{
7931 #2\l_@@_inclasssign_title_tl#3
7932 }
7933 }

```

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

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

```

7934 \newcommand\assignment@number{
7935 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
7936 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7937 \arabic{assignment}
7938 } {
7939 \int_use:N \l_@@_assign_number_int
7940 }
7941 }{
7942 \int_use:N \l_@@_inclasssign_number_int
7943 }
7944 }

```

(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.

```

7945 \newenvironment{assignment}[1][ ]{
7946 \_@@_assignment_args:n { #1 }
7947 %\sref@target
7948 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7949 \global\stepcounter{assignment}
7950 }{
7951 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7952 }
7953 \setcounter{problem}{0}
7954 \renewcommand\prob@label[1]{\assignment@number.##1}
7955 \def\current@section@level{\document@hwexamtype}
7956 %\sref@label{id}{\document@hwexamtype \thesection}
7957 \begin{@assignment}
7958 }{
7959 \end{@assignment}
7960 }

```

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

```

7961 \def\ass@title{
7962 {\protect\document@hwexamtype}\arabic{assignment}
7963 \assignment@title{\;(\;)\;}\; -- \given@due{}\;}
7964 }
7965 \ifmultiple
7966 \newenvironment{@assignment}{
7967 \bool_if:NTF \l_@@_assign_loadmodules_bool {
7968 \begin{sfragment}[loadmodules]{\ass@title}
7969 }{
7970 \begin{sfragment}{\ass@title}
7971 }
7972 }{
7973 \end{sfragment}
7974 }

```

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

```

7975 \else
7976 \newenvironment{@assignment}{
7977 \begin{center}\bf
7978 \Large@title\strut\
7979 \document@hwexamtype}\arabic{assignment}\assignment@title{\;(\;)\;}\;
7980 \large\given@due{--\;}\;{\;--}
7981 \end{center}
7982 }{}
7983 \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.

```

7984 \keys_define:nn { hwexam / inclassignment } {
7985 %id .str_set_x:N = \l_@@_assign_id_str,
7986 number .int_set:N = \l_@@_inclassign_number_int,
7987 title .tl_set:N = \l_@@_inclassign_title_tl,
7988 type .tl_set:N = \l_@@_inclassign_type_tl,
7989 given .tl_set:N = \l_@@_inclassign_given_tl,
7990 due .tl_set:N = \l_@@_inclassign_due_tl,
7991 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
7992 }
7993 \cs_new_protected:Nn \_@@_inclassignment_args:n {
7994 \int_set:Nn \l_@@_inclassign_number_int {-1}
7995 \tl_clear:N \l_@@_inclassign_title_tl
7996 \tl_clear:N \l_@@_inclassign_type_tl
7997 \tl_clear:N \l_@@_inclassign_given_tl
7998 \tl_clear:N \l_@@_inclassign_due_tl
7999 \str_clear:N \l_@@_inclassign_mhrepos_str
8000 \keys_set:nn { hwexam / inclassignment }{ #1 }
8001 }
8002 \_@@_inclassignment_args:n {}
8003
8004 \newcommand\inputassignment[2][{}]{

```

```

8005 \_@@_inclassassignment_args:n { #1 }
8006 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8007   \input{#2}
8008 }{
8009   \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8010     \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8011   }
8012 }
8013 \_@@_inclassassignment_args:n {}
8014 }
8015 \newcommand\includeassignment[2][]{
8016   \newpage
8017   \inputassignment[#1]{#2}
8018 }

```

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

40.4 Typesetting Exams

\quizheading

```

8019 \ExplSyntaxOff
8020 \newcommand\quizheading[1]{%
8021   \def\@tas{#1}%
8022   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8023   \ifx\@tas\@empty\else%
8024     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8025   \fi%
8026 }
8027 \ExplSyntaxOn

```

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

\testheading

```

8028
8029 \def\hwexamheader{\input{hwexam-default.header}}
8030
8031 \def\hwexamminutes{
8032   \tl_if_empty:NTF \testheading@duration {
8033     {\testheading@min}~\hwexam@minutes@kw
8034   }{
8035     \testheading@duration
8036   }
8037 }
8038
8039 \keys_define:nn { hwexam / testheading } {
8040   min .tl_set:N = \testheading@min,
8041   duration .tl_set:N = \testheading@duration,
8042   reqpts .tl_set:N = \testheading@reqpts,
8043   tools .tl_set:N = \testheading@tools
8044 }
8045 \cs_new_protected:Nn \_@@_testheading_args:n {
8046   \tl_clear:N \testheading@min
8047   \tl_clear:N \testheading@duration

```



```

8048 \tl_clear:N \testheading@reqpts
8049 \tl_clear:N \testheading@tools
8050 \keys_set:nn { hwexam / testheading }{ #1 }
8051 }
8052 \newenvironment{testheading}[1][]{
8053 \_@@_testheading_args:n{ #1 }
8054 \newcount\check@time\check@time=\testheading@min
8055 \advance\check@time by -\theassignment@totalmin
8056 \newif\if@bonuspoints
8057 \tl_if_empty:NTF \testheading@reqpts {
8058 \@bonuspointsfalse
8059 }{
8060 \newcount\bonus@pts
8061 \bonus@pts=\theassignment@totalpts
8062 \advance\bonus@pts by -\testheading@reqpts
8063 \edef\bonus@pts{\the\bonus@pts}
8064 \@bonuspointstrue
8065 }
8066 \edef\check@time{\the\check@time}
8067
8068 \makeatletter\hwexamheader\makeatother
8069 }{
8070 \newpage
8071 }

```

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

\testspace

```

8072 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8073 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8074 \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.

```

8075 <@=problems>
8076 \renewcommand\@problem[3]{
8077 \stepcounter{assignment@probs}
8078 \def\__problemspts{#2}
8079 \ifx\__problemspts\@empty\else
8080 \addtocounter{assignment@totalpts}{#2}
8081 \fi
8082 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8083 \xdef\correction@probs{\correction@probs & #1}%
8084 \xdef\correction@pts{\correction@pts & #2}
8085 \xdef\correction@reached{\correction@reached &}

```

```

8086 }
8087 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

8088 \newcounter{assignment@probs}
8089 \newcounter{assignment@totalpts}
8090 \newcounter{assignment@totalmin}
8091 \def\correction@probs{\correction@probs@kw}
8092 \def\correction@pts{\correction@pts@kw}
8093 \def\correction@reached{\correction@reached@kw}
8094 \stepcounter{assignment@probs}
8095 \newcommand\correction@table{
8096 \resizebox{\textwidth}{!}{%
8097 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8098 &\multicolumn{\theassignment@probs}{c|}|%|
8099 {\footnotesize\correction@forgrading@kw} &\\ \hline
8100 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8101 \correction@pts & \theassignment@totalpts & \\ \hline
8102 \correction@reached & & \[.7cm]\hline
8103 \end{tabular}}
8104 </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).