

The sTeX3 Package Collection *

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

2022-05-09

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.0 (last revised 2022-05-09)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	Minimal Setup for the PDF-only Workflow	3
2.1.2	GIT-based Setup for the sTeX Development Version	3
2.1.3	sTeX Archives (Manual Setup)	4
2.1.4	The sTeX IDE	4
2.1.5	Manual Setup for Active Documents and Knowledge Management Services	4
2.2	A First sTeX Document	5
2.2.1	OMDoc/xhtml Conversion	8
3	Creating sTeX Content	10
3.1	How Knowledge is Organized in sTeX	10
3.2	sTeX Archives	11
3.2.1	The Local MathHub-Directory	11
3.2.2	The Structure of sTeX Archives	11
3.2.3	MANIFEST.MF-Files	12
3.2.4	Using Files in sTeX Archives Directly	13
3.3	Module, Symbol and Notation Declarations	14
3.3.1	The smodule-Environment	14
3.3.2	Declaring New Symbols and Notations	16
	Operator Notations	19
3.3.3	Argument Modes	19
	Mode-b Arguments	20
	Mode-a Arguments	20
	Mode-B Arguments	22
3.3.4	Type and Definiens Components	22
3.3.5	Precedences and Automated Bracketing	23
3.3.6	Variables	25
3.3.7	Variable Sequences	26
3.4	Module Inheritance and Structures	28
3.4.1	Multilinguality and Translations	28
3.4.2	Simple Inheritance and Namespaces	29
3.4.3	The mathstructure Environment	31
3.4.4	The copymodule Environment	34
3.4.5	The interpretmodule Environment	35
3.5	Primitive Symbols (The sTeX Metatheory)	35
4	Using sTeX Symbols	36
4.1	\symref and its variants	36
4.2	Marking Up Text and On-the-Fly Notations	37
4.3	Referencing Symbols and Statements	39

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

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

28	sTeX-Module Inheritance Implementation	134
28.1	SMS Mode	134
28.2	Inheritance	138
29	sTeX-Symbols Implementation	144
29.1	Symbol Declarations	144
29.2	Notations	151
29.3	Variables	160
30	sTeX-Terms Implementation	166
30.1	Symbol Invocations	166
30.2	Terms	173
30.3	Notation Components	177
30.4	Variables	179
30.5	Sequences	181
31	sTeX-Structural Features Implementation	182
31.1	Imports with modification	183
31.2	The feature environment	191
31.3	Structure	191
32	sTeX-Statements Implementation	201
32.1	Definitions	201
32.2	Assertions	206
32.3	Examples	210
32.4	Logical Paragraphs	212
33	The Implementation	218
33.1	Proofs	218
33.2	Justifications	229
34	sTeX-Others Implementation	230
35	sTeX-Metatheory Implementation	231
36	Tikzinput Implementation	234
37	document-structure.sty Implementation	237
37.1	Package Options	237
37.2	Document Structure	238
37.3	Front and Backmatter	242
37.4	Global Variables	244
38	NotesSlides – Implementation	245
38.1	Class and Package Options	245
38.2	Notes and Slides	247
38.3	Header and Footer Lines	251
38.4	Frame Images	253
38.5	Colors and Highlighting	254
38.6	Sectioning	255
38.7	Excursions	257

39 The Implementation	259
39.1 Package Options	259
39.2 Problems and Solutions	260
39.3 Multiple Choice Blocks	267
39.4 Including Problems	268
39.5 Reporting Metadata	270
40 Implementation: The hwexam Package	272
40.1 Package Options	272
40.2 Assignments	273
40.3 Including Assignments	276
40.4 Typesetting Exams	277
40.5 Leftovers	279
41 References	280

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 $\text{\texttt{T\text{E}XLive}}$ 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 $\text{\texttt{T\text{E}XLive}}$ via a package manager or the $\text{\texttt{T\text{E}XLive}}$ manager **$\text{\texttt{tlmgr}}$** .

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

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

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

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

and keep it updated by pulling updates via `git pull` in the cloned $\text{\texttt{sTeX}}$ directory. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

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

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

2.1.3 sTeX Archives (Manual Setup)

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

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

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

Note that sTeX archives often depend on other archives, thus you should be prepared to clone these as well – e.g. if `pdflatex` reports missing files. To make sure that sTeX too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local MathHub-directory (see [section 3.2](#)).

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

2.1.4 The sTeX IDE

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

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

Foregoing on the sTeX IDE, we will need several additional (on top of the minimal setup above) pieces of software; namely:

- **The Mmt System** available [here](#)². We recommend following the setup routine documented [here](#).

Following the setup routine (Step 3) will entail designating a **MathHub**-directory on your local file system, where the MMT system will look for sTeX/MMT content archives.

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

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

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

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

2.2 A First \LaTeX Document

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

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

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 (*<language>**) Languages to load with the `babel` package.

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

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

image (*<boolean>*) passed on to `tikzinput`.

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

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

1. sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
2. These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
3. Modules contain sTeX **symbol declarations**, introduced via `\symdecl{symbolname}`, `\symdef{symbolname}` and some other constructions. Most symbols have a *notation* that can be used via a *semantic macro* `\symbolname` generated by symbol declarations.
4. sTeX **expressions** finally are built up from usages of semantic macros.

↖M↗

→M→

↘T↙

- sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.
- sTeX modules correspond to OMDoc/MMT *theories*. `\importmodules` (and

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

similar constructions) induce MMT `includes` and other *theory morphisms*, thus giving rise to a *theory graph* in the OMDOC sense [RK13].

- Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ expressions are converted to OMDOC/MMT terms, which use the abstract syntax (and XML encoding) of OPENMATH [Bus+04].

3.2 $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Archives

3.2.1 The Local MathHub-Directory

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

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

1. If the $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.
4. Finally, if all else fails, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ will look for a file `~/.stex/mathhub.path`. If this file exists, $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ will assume that it contains the path to the local MathHub-directory. This method is recommended on systems where it is difficult to set environment variables.

3.2.2 The Structure of $\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ Archives

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

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where $\text{\texttt{S\TeX}}$ will look for files included via `\libinput`.

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

We recommend the following additional directory structure in the `source`-folder of an $\text{\texttt{S\TeX}}$ archive:

- `/source/mod/` – individual $\text{\texttt{S\TeX}}$ modules, containing symbol declarations, notations, and `\begin{sparagraph}[type=symdoc,for=...]` environments for “encyclopaedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputrefing` snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.³

3.2.3 MANIFEST.MF-Files

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

```
id: smglom/calculus
source-base: http://mathhub.info/smgom/calculus
narration-base: http://mathhub.info/smgom/calculus
dependencies: smglom/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

Many of these are in fact ignored by $\text{\texttt{S\TeX}}$, but some are important:

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

`source-base` or

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

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

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

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

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

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

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

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

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

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

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

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

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

$\text{\texttt{\backslash libinput}}$	$\text{\texttt{\backslash libinput}}\{\text{some/file}\}$ searches for a file <code>some/file</code> in
---------------------------------------	---

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

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

$\text{\texttt{\backslash libinput}}$ will throw an error if *no* candidate for `some/file` is found.

`\libusepackage` `\libusepackage`[package-options]{some/file} searches for a file `some/file.sty` in the same way that `\libinput` does, but will call `\usepackage`[package-options]{path/to/some/file} instead of `\input`.
`\libusepackage` throws an error if not *exactly one* candidate for `some/file` is found.

Remark 3.2.1:

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

```

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

```

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

`smodule` A new module is declared using the basic syntax

```
\begin{smodule}[options]{ModuleName}...\end{smodule}.
```

A module is required to declare any new formal content such as symbols or notations (but not variables, which may be introduced anywhere).

The `smodule`-environment takes several keyword arguments, all of which are optional:

`title` (*<token list>*) to display in customizations.

`type` (*<string>**) for use in customizations.

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

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

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

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

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

`creators` ($\langle string \rangle^*$) names of the creators.
`contributors` ($\langle string \rangle^*$) names of contributors.
`srccite` ($\langle string \rangle$) a source citation for the content of this module.

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

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

Example 1

Input:

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

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

`\comp`

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

We can introduce a new notation `highlight` for `\binarysymbol` that fixes this flaw, which we can subsequently use with `\binarysymbol[highlight]`:

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

```
1 \symdef{newbinarysymbol}[h1,args=2]
2   {\comp{\text{1.: }}#1\comp{\text{; 2.: }}#2}
3 $\newbinarysymbol{a}{b}$
```

Output:

1.: *a*; 2.: *b*

We just declared a new symbol `newbinarysymbol` with `args=2` and immediately provided it with a notation with identifier `h1`. Since `h1` is the *first* (and so far, only) notation supplied for `newbinarysymbol`, using `\newbinarysymbol` without optional argument defaults to this notation.

But one man's meat is another man's poison: it is very subjective what the "default notation" of an operator should be. Different communities have different practices. For instance, the complex unit is written as *i* in Mathematics and as *j* in electrical engineering.

So to allow modular specification and facilitate re-use of document fragments \LaTeX allows to re-set notation defaults.

$\backslash\text{setnotation}$

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

newbinarysymbol is also occasionally written $\text{a:} \cdot \text{; b:} \cdot$

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

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.
- \rightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \rightarrow 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  $\seqa!$  is  $\seqa{i}$ .

```

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

```

Output:

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

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

Example 23

Input:

```

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

```

Output:

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

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

If a module `Foo` exists in e.g. english in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key

then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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



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



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

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

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

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

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

`\STEXexport`

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



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

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T_EX



group, such as `\def` or `\let`.

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

A `monoid` is...

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

Example 25

Input:

```

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

```

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

\instantiate

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

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

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

$\hookrightarrow M$ from (and correspond to) the constants in `<name>-structure`.
 $\hookrightarrow M$ `\instantiate` generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned based on the respective key-value-list.
 $\rightsquigarrow T$

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

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

`\varinstantiate`

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

This allows us to do things like:

Example 27

Input:

```

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

```

Output:

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

.

and

Example 28

Input:

```

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

```

Output:

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

.

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

.

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 31

Input:

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

Output:

3.5 Primitive Symbols (The $\text{\texttt{STEX}}$ 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
 \rightarrow T \rightarrow `<OMV name="m"/>` as arguments.



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

\arg

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

`\arg` also allows us to switch the order of arguments around and “hide” arguments: For example, `\arg[3]{<code>}` signifies that `<code>` represents the *third* argument to the current operator, and `\arg*[i]{<code>}` signifies that `<code>` represents the *i*th argument, but it should not produce any output (it is exported in the `xhtml` however, so that MMT and other systems can pick up on it).⁵

Example 36

Input:

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

Output:

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

adding k yields...

Note that since the second `\arg` has no explicit argument number, it automatically represents the first not-yet-given argument – i.e. in this case the first one.⁶

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

Example 37

Input:

```
1 Given  $\text{\addition{\svar{n}}{\svar{m}}}$ , 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>}{<text>}` tests the content of the global variable `<vname>`, only if (after expansion) it is equal to `<val>`, the conditional text `<text>` is formatted.

7.3 Slides and Course Notes

The `notesslides` document class is derived from `beamer.cls` [Tana], it adds a “notes version” for course notes that is more suited to printing than the one supplied by `beamer.cls`.

The `notesslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the `STEX` and `OMDOC`. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `notesslides` package displays the slides as such in the course notes to give students a visual anchor into the slide presentation in the course (and to distinguish the different writing styles in slides and course notes).

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

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



Note that it is essential to start and end the `notes` environment at the start of the line – in particular, there may not be leading blanks – else L^AT_EX becomes confused and throws error messages that are difficult to decipher.

By interleaving the `frame` and `note` environments, we can build course notes as shown here:

```

1 \ifnotes\maketitle\else
2 \frame[noframenumbering]\maketitle\fi
3
4 \begin{note}
5   We start this course with ...
6 \end{note}
7
8 \begin{frame}
9   \frametitle{The first slide}
10  ...
11 \end{frame}
12 \begin{note}
13   ... and more explanatory text
14 \end{note}
15
16 \begin{frame}
17   \frametitle{The second slide}
18   ...
19 \end{frame}
20 ...

```

`\ifnotes`

Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notestru` or `\notesfalse` is strongly discouraged however.

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



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



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

`\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

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

`\setlicensing`

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

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

`\frameimage`
`\mhframeimage`

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

The `\mhframeimage` macro is a variant of `\frameimage` with repository support. Instead of writing

```
1 \frameimage{\MathHub{fooMH/bar/source/baz/foobar}}
```

we can simply write (assuming that `\MathHub` is defined as above)

```
1 \mhframeimage[fooMH/bar]{baz/foobar}
```

Note that the `\mhframeimage` form is more semantic, which allows more advanced document management features in `MathHub`.

If `baz/foobar` is the “current module”, i.e. if we are on the `MathHub` path `...MathHub/fooMH/bar...`, then stating the repository in the first optional argument is redundant, so we can just use

```
1 \mhframeimage{baz/foobar}
```

`\textwarning`

The `\textwarning` macro generates a warning sign: 

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

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

`\excursion`

The `\excursion{⟨ref⟩}{⟨path⟩}{⟨text⟩}` is syntactic sugar for

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

`\activateexcursion`
`\printexcursion`
`\excursionref`

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

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

`\excursiongroup`

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

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



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

7.4 Representing Problems and Solutions

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

Finally, the `problem` package facilitates the management of problems in small files, so that problems can be re-used in multiple environment.

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

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

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

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

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

Example 40

Input:

```

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

```

Output:

Problem 7.4.1 (Fitting Elephants)
 How many Elephants can you fit into a Volkswagen beetle?

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

`\ifsolutions`

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

`mcb` Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc` macro.

`\mcc`

`\mcc[⟨keyvals⟩]{⟨text⟩}` takes an optional key/value argument `⟨keyvals⟩` for choice meta-data and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

- `T` for true answers, `F` for false ones,
- `Ttext` the verdict for true answers, `Ftext` for false ones, and
- `feedback` for a short feedback text given to the student.

If we start the solutions, then we get

Example 41

Input:

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

2022-05-09

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

	To be used for grading, do not write here											
prob.	7.4.1	7.4.2	7.4.3	1.1	2.1	2.2	2.3	3.1	3.2	3.3	Sum	grade
total	10			4	4	6	6	4	4	2	40	
reached												

good luck

EdN:13

13

\inputassignment

The `\inputassignment` macro can be used to input an assignment from another file. It takes an optional `KeyVal` argument and a second argument which is a path to the file containing the problem (the macro assumes that there is only one `assignment` environment in the included file). The keys `number`, `title`, `type`, `given`, and `due` are just as for the `assignment` environment and (if given) overwrite the ones specified in the `assignment` environment in the included file.

¹³EDNOTE: MK: The first three “problems” come from the stex examples above, how do we get rid of this?

Part II
Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

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

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

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

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

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

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

Annotates the HTML generated by $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>	
--	--

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

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

<code>\stex_set_current_repository:n</code>	
---	--

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

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

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

Chapter 10

STEX-References

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

10.1 Macros and Environments

\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:

 $\backslash\text{c_stex_module_}\langle URI \rangle_prop$

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

 $\backslash\text{c_stex_module_}\langle URI \rangle_code$

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

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

The names of all constants declared in the module

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

<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 \STEX 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 \STEX 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

STEX-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

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7
8 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
9 \ProcessOptions
10
11 \bool_set_true:N \c_stex_document_class_bool
12
13 \RequirePackage{stex}
14
15 \stex_html_backend:TF {
16   \LoadClass{article}
17 }{
18   \LoadClass[border=1px,varwidth,crop=false]{standalone}
19   \setlength\textwidth{15cm}
20 }
21 \RequirePackage{standalone}
22
23
24 \clist_if_empty:NT \c_stex_languages_clist {
25   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
26   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
27   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
28   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
29     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
30       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
```

```

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

```

24.2 Preliminaries

```

44 <*package>
45
46 %%%%%%%%% basics.dtx %%%%%%%%%
47
48 \RequirePackage{expl3,l3keys2e,ltxcmds}
49 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
50
51 \bool_if_exist:NF \c_stex_document_class_bool {
52   \bool_set_false:N \c_stex_document_class_bool
53   \RequirePackage{standalone}
54 }
55
56 \message{^^J
57   *****^^J
58   *~This~is~sTeX~version~3.1.0*~^^J
59   *****^^J
60 ^^J}
61
62 %\RequirePackage{morewrites}
63 %\RequirePackage{amsmath}
64

```

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

```

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

```

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

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

116 `<@=stex_annotate>`

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

117 `\tl_new:N \l_stex_html_arg_tl`

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

`_stex_html_checkempty:n`

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

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

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

`\stex_if_do_html:TF`

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

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

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

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

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

`\stex_annotate:bnx`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

146     \def\stex@backend{latexml}
147   \else
148     \def\stex@backend{pdflatex}
149   \fi
150 \fi
151 }
152 \input{stex-backend-\stex@backend.cfg}
153
154 \newif\ifstexhtml
155 \stex_html_backend:TF\stexhtmltrue\stexhtmlfalse
156

```

(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

```

157 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

183 \cs_new_protected:Nn \stex_set_language:Nn {
184   \str_set:Nx \l_tmpa_str {#2}
185   \prop_get:NoNT \c_stex_languages_prop \l_tmpa_str #1 {
186     \ifx\@onlypreamble\@notprerr
187       \ltx@ifpackageloaded{babel}{

```

```

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

```

24.6 Persistence

```

236 <@@=stex_persist>
237 \bool_if:NTF \c_stex_persist_mode_bool {

```

```

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

```

24.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

258 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
259 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
260 \def#1{
261 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
262 }
263 }

```

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

\stex_reactivate_macro:N

```

264 \cs_new_protected:Nn \stex_reactivate_macro:N {
265 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
266 }

```

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

\ignorespacesandpars

```

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

```



```

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

```

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

`\MMTrule`

```

334 \NewDocumentCommand \MMTrule {m m}{
335   \seq_set_split:Nnn \l_tmpa_seq , {#2}
336   \int_zero:N \l_tmpa_int
337   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
338     \seq_if_empty:NF \l_tmpa_seq {
339       $\seq_map_inline:Nn \l_tmpa_seq {
340         \int_incr:N \l_tmpa_int
341         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
342       }$
343     }
344   }
345 }
346
347 \NewDocumentCommand \MMTinclude {m}{
348   \stex_annotate_invisible:nnn{import}{#1}{ }
349 }
350
351 \tl_new:N \g_stex_document_title
352 \cs_new_protected:Npn \STEXtitle #1 {
353   \tl_if_empty:NT \g_stex_document_title {
354     \tl_gset:Nn \g_stex_document_title { #1 }
355   }
356 }
357 \cs_new_protected:Nn \stex_document_title:n {
358   \tl_if_empty:NT \g_stex_document_title {
359     \tl_gset:Nn \g_stex_document_title { #1 }
360     \stex_annotate_invisible:n{\noindent
361       \stex_annotate:nnn{doctitle}{ }{ #1 }
362     }
363   }
364 }
365 \AtBeginDocument {
366   \let \STEXtitle \stex_document_title:n
367   \tl_if_empty:NF \g_stex_document_title {
368     \stex_annotate_invisible:n{\noindent
369       \stex_annotate:nnn{doctitle}{ }{ \g_stex_document_title }
370     }
371   }
372   \let \stex_maketitle \maketitle
373   \def \maketitle {
374     \tl_if_empty:NF \@title {
375       \exp_args:No \stex_document_title:n \@title
376     }
377     \stex_maketitle:
378   }
379 }
380
381 \cs_new_protected:Nn \stex_par: {
382   \mode_if_vertical:F{
383     \if@minipage\else\if@nobreak\else\par\fi\fi
384   }

```

385 }

386

387 \langle /package \rangle

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

Chapter 25

STEX -MathHub Implementation

```
388 <*package>
389
390 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
391
392 <@@=stex_path>
393
394   Warnings and error messages
395   \msg_new:nnn{stex}{error/norepository}{
396     No~archive~#1~found~in~#2
397   }
398   \msg_new:nnn{stex}{error/notinarchive}{
399     Not~currently~in~an~archive,~but~\detokenize{#1}~
400     needs~one!
401   }
402   \msg_new:nnn{stex}{error/nofile}{
403     \detokenize{#1}~could~not~find~file~#2
404   }
405   \msg_new:nnn{stex}{error/twofiles}{
406     \detokenize{#1}~found~two~candidates~for~#2
407   }
```

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`

```
406 \cs_new_protected:Nn \stex_path_from_string:Nn {
407   \str_set:Nx \l_tmpa_str { #2 }
408   \str_if_empty:NTF \l_tmpa_str {
409     \seq_clear:N #1
410   }{
411     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
412     \sys_if_platform_windows:T{
413       \seq_clear:N \l_tmpa_tl
```

```

414     \seq_map_inline:Nn #1 {
415       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
416       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
417     }
418     \seq_set_eq:NN #1 \l_tmpa_tl
419   }
420   \stex_path_canonicalize:N #1
421 }
422 }
423

```

(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`

```

424 \cs_new_protected:Nn \stex_path_to_string:NN {
425   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
426 }
427
428 \cs_new:Nn \stex_path_to_string:N {
429   \seq_use:Nn #1 /
430 }

```

(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`

```

431 \str_const:Nn \c__stex_path_dot_str {.}
432 \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.

```

433 \cs_new_protected:Nn \stex_path_canonicalize:N {
434   \seq_if_empty:NF #1 {
435     \seq_clear:N \l_tmpa_seq
436     \seq_get_left:NN #1 \l_tmpa_tl
437     \str_if_empty:NT \l_tmpa_tl {
438       \seq_put_right:Nn \l_tmpa_seq {}
439     }
440     \seq_map_inline:Nn #1 {
441       \str_set:Nn \l_tmpa_tl { ##1 }
442       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
443         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
444           \seq_if_empty:NNTF \l_tmpa_seq {
445             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
446               \c__stex_path_up_str
447             }
448           }{
449             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
450             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
451               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
452                 \c__stex_path_up_str
453               }
454             }{

```

```

455         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
456     }
457 }
458 }{
459     \str_if_empty:NF \l_tmpa_tl {
460         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
461     }
462 }
463 }
464 }
465 \seq_gset_eq:NN #1 \l_tmpa_seq
466 }
467 }

```

(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}`

```

468 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
469     \seq_if_empty:NTF #1 {
470         \prg_return_false:
471     }{
472         \seq_get_left:NN #1 \l_tmpa_tl
473         \sys_if_platform_windows:TF{
474             \str_if_in:NnTF \l_tmpa_tl {:}{
475                 \prg_return_true:
476             }{
477                 \prg_return_false:
478             }
479         }{
480             \str_if_empty:NTF \l_tmpa_tl {
481                 \prg_return_true:
482             }{
483                 \prg_return_false:
484             }
485         }
486     }
487 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

488 \str_new:N\l_stex_kpsewhich_return_str
489 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
490     \catcode'\ =12
491     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
492     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
493     \endgroup
494     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
495     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
496 }

```

(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`

```

497 \sys_if_platform_windows:TF{
498   \begingroup\escapechar=-1\catcode'\=12
499   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
500   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
501   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
502   }}{
503   \stex_kpsewhich:n{-var-value~PWD}
504   }
505
506 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
507 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
508 \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

509 `<@@=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

```

510 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

511 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
512 \stex_path_from_string:Nn \c_stex_mainfile_seq
513   \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`

```

514 \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`

```

515 \cs_new_protected:Nn \stex_filestack_push:n {
516   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
517   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
518     \stex_path_from_string:Nn\g_stex_currentfile_seq{
519       \c_stex_pwd_str/#1
520     }

```

```

521 }
522 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
523 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
524 }

```

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

`\stex_filestack_pop:`

```

525 \cs_new_protected:Nn \stex_filestack_pop: {
526   \seq_if_empty:NF\g__stex_files_stack{
527     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
528   }
529   \seq_if_empty:NTF\g__stex_files_stack{
530     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
531   }{
532     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
533     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
534   }
535 }

```

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

Hooks for the current file:

```

536 \AddToHook{file/before}{
537   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
538 }
539 \AddToHook{file/after}{
540   \stex_filestack_pop:
541 }

```

25.4 MathHub Repositories

```

542 <@=stex_mathhub>

```

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

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

```

543 \str_if_empty:NTF\mathhub{
544   \sys_if_platform_windows:TF{
545     \begingroup\escapechar=-1\catcode'\=12
546     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
547     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
548     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
549   }{
550     \stex_kpsewhich:n{-var-value-MATHHUB}
551   }
552   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
553 }
554 \str_if_empty:NT \c_stex_mathhub_str {
555   \sys_if_platform_windows:TF{
556     \begingroup\escapechar=-1\catcode'\=12
557     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
558     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
559     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
560   }{

```



```

561     \stex_kpsewhich:n{-var-value~HOME}
562   }
563   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
564     \begingroup\escapechar=-1\catcode'\=12
565     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
566     \sys_if_platform_windows:T{
567       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{c_backslash_str}/
568     }
569     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
570     \endgroup
571     \ior_close:N \l_tmpa_ior
572   }
573 }
574 \str_if_empty:NTF\c_stex_mathhub_str{
575   \msg_warning:nn{stex}{warning/nomathhub}
576 }{
577   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
578   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
579 }
580 }{
581   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
582   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
583     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
584       \c_stex_pwd_str/\mathhub
585     }
586   }
587   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
588   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
589 }

```

(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

```

590 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
591   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
592     \str_set:Nx \l_tmpa_str { #1 }
593     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
594     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
595     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
596     \__stex_mathhub_find_manifest:N \l_tmpa_seq
597     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
598       \msg_error:nnxx{stex}{error/norepository}{#1}{
599         \stex_path_to_string:N \c_stex_mathhub_str
600       }
601     } {
602       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
603     }
604   }
605 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```
606 \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`:

```
607 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
608   \seq_set_eq:NN\l_tmpa_seq #1
609   \bool_set_true:N\l_tmpa_bool
610   \bool_while_do:Nn \l_tmpa_bool {
611     \seq_if_empty:NTF \l_tmpa_seq {
612       \bool_set_false:N\l_tmpa_bool
613     }{
614       \file_if_exist:nTF{
615         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
616       }{
617         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
618         \bool_set_false:N\l_tmpa_bool
619       }{
620         \file_if_exist:nTF{
621           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
622         }{
623           \seq_put_right:Nn\l_tmpa_seq{META-INF}
624           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
625           \bool_set_false:N\l_tmpa_bool
626         }{
627           \file_if_exist:nTF{
628             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
629           }{
630             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
631             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
632             \bool_set_false:N\l_tmpa_bool
633           }{
634             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
635           }
636         }
637       }
638     }
639   }
640   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
641 }
```

(End definition for `__stex_mathhub_find_manifest:N`.)

`\c__stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```
642 \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:

```
643 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
644   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
645   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
646   \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
```

```

647 \str_set:Nn \l_tmpa_str {##1}
648 \exp_args:NNoo \seq_set_split:Nnn
649   \l_tmpb_seq \c_colon_str \l_tmpa_str
650 \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
651   \exp_args:NNe \str_set:Nn \l_tmpb_tl {
652     \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
653   }
654   \exp_args:No \str_case:nnTF \l_tmpa_tl {
655     {id} {
656       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
657       { id } \l_tmpb_tl
658     }
659     {narration-base} {
660       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
661       { narr } \l_tmpb_tl
662     }
663     {url-base} {
664       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
665       { docurl } \l_tmpb_tl
666     }
667     {source-base} {
668       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
669       { ns } \l_tmpb_tl
670     }
671     {ns} {
672       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
673       { ns } \l_tmpb_tl
674     }
675     {dependencies} {
676       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
677       { deps } \l_tmpb_tl
678     }
679   }{}{}
680 }{}
681 }
682 \ior_close:N \c__stex_mathhub_manifest_ior
683 \stex_persist:x {
684   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
685     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
686   }
687 }
688 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

689 \cs_new_protected:Nn \stex_set_current_repository:n {
690   \stex_require_repository:n { #1 }
691   \prop_set_eq:Nc \l_stex_current_repository_prop {
692     c_stex_mathhub_#1_manifest_prop
693   }
694 }

```

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

`\stex_require_repository:n`

```

695 \cs_new_protected:Nn \stex_require_repository:n {
696   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
697     \stex_debug:nn{mathhub}{Opening~archive:~#1}
698     \__stex_mathhub_do_manifest:n { #1 }
699   }
700 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

701 %\prop_new:N \l_stex_current_repository_prop
702 \bool_if:NF \c_stex_persist_mode_bool {
703   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
704   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
705     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
706   } {
707     \__stex_mathhub_parse_manifest:n { main }
708     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
709     \l_tmpa_str
710     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
711     \c_stex_mathhub_main_manifest_prop
712     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
713     \stex_debug:nn{mathhub}{Current~repository:~
714     \prop_item:Nn \l_stex_current_repository_prop {id}
715   }
716 }
717 }

```

(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.

```

718 \cs_new_protected:Nn \stex_in_repository:nn {
719   \str_set:Nx \l_tmpa_str { #1 }
720   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
721   \str_if_empty:NTF \l_tmpa_str {
722     \prop_if_exist:NTF \l_stex_current_repository_prop {
723       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
724       \exp_args:Ne \l_tmpa_cs{
725         \prop_item:Nn \l_stex_current_repository_prop { id }
726       }
727     }{
728       \l_tmpa_cs{}
729     }
730   }{
731     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
732     \stex_require_repository:n \l_tmpa_str
733     \str_set:Nx \l_tmpa_str { #1 }
734     \exp_args:Nne \use:nn {
735       \stex_set_current_repository:n \l_tmpa_str
736       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
737     }{
738       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

739     \prop_if_exist:NTF \l_stex_current_repository_prop {
740       \prop_item:Nn \l_stex_current_repository_prop { id } :~
741       \meaning\l_stex_current_repository_prop
742     }{
743       no~repository
744     }
745   }
746   \prop_if_exist:NTF \l_stex_current_repository_prop {
747     \stex_set_current_repository:n {
748       \prop_item:Nn \l_stex_current_repository_prop { id }
749     }
750   }{
751     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
752   }
753 }
754 }
755 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

756 \def \mhpath #1 #2 {
757   \exp_args:Ne \tl_if_empty:nTF{#1}{
758     \c_stex_mathhub_str /
759     \prop_item:Nn \l_stex_current_repository_prop { id }
760     / source / #2
761   }{
762     \c_stex_mathhub_str / #1 / source / #2
763   }
764 }

```

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

`\inputref`

`\mhinput`

```

765 \newif \ifinputref \inputreffalse
766
767 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
768   \stex_in_repository:nn {#1} {
769     \ifinputref
770       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
771     \else
772       \inputreftrue
773       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
774       \inputreffalse
775     \fi
776   }
777 }
778 \NewDocumentCommand \mhinput { 0{} m }{
779   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
780 }
781

```

```

782 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
783   \stex_in_repository:nn {#1} {
784     \stex_html_backend:TF {
785       \str_clear:N \l_tmpa_str
786       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
787         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
788       }
789
790       \tl_if_empty:nTF{ ##1 }{
791         \IfFileExists{#2}{
792           \stex_annotate_invisible:nnn{inputref}{
793             \l_tmpa_str / #2
794           }{}
795         }{
796           \input{#2}
797         }
798       }{
799         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
800           \stex_annotate_invisible:nnn{inputref}{
801             \l_tmpa_str / #2
802           }{}
803         }{
804           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
805         }
806       }
807
808     }{
809       \begingroup
810       \inputreftrue
811       \tl_if_empty:nTF{ ##1 }{
812         \input{#2}
813       }{
814         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
815       }
816       \endgroup
817     }
818   }
819 }
820 \NewDocumentCommand \inputref { 0{} m}{
821   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
822 }

```

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

`\addmhbibresource`

```

823 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
824   \stex_in_repository:nn {#1} {
825     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
826   }
827 }
828 \newcommand\addmhbibresource[2][]{
829   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
830 }

```

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

`\libinput`

```
831 \cs_new_protected:Npn \libinput #1 {
832   \prop_if_exist:NF \l_stex_current_repository_prop {
833     \msg_error:nnn{stex}{error/notinarchive}\libinput
834   }
835   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
836     \msg_error:nnn{stex}{error/notinarchive}\libinput
837   }
838   \seq_clear:N \l__stex_mathhub_libinput_files_seq
839   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
840   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
841
842   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
843     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
844     \IfFileExists{ \l_tmpa_str }{
845       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
846     }{}
847     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
848     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
849   }
850
851   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
852   \IfFileExists{ \l_tmpa_str }{
853     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
854   }{}
855
856   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
857     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
858   }{
859     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
860       \input{ ##1 }
861     }
862   }
863 }
```

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

`\libusepackage`

```
864 \NewDocumentCommand \libusepackage {0{ } m} {
865   \prop_if_exist:NF \l_stex_current_repository_prop {
866     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
867   }
868   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
869     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
870   }
871   \seq_clear:N \l__stex_mathhub_libinput_files_seq
872   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
873   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
874
875   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
876     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
877     \IfFileExists{ \l_tmpa_str.sty }{
878       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
879     }{}
880   }
```

```

880 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
881 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
882 }
883
884 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
885 \IfFileExists{ \l_tmpa_str.sty }{
886 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
887 }{}
888
889 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
890 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
891 }{
892 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
893 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
894 \usepackage[#1]{ ##1 }
895 }
896 }{
897 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
898 }
899 }
900 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

901
902 \AddToHook{begindocument}{
903 \ltx@ifpackageloaded{graphicx}{
904 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
905 \newcommand\mhgraphics[2][]{\%
906 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
907 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
908 \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
909 }{}

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

910 \ltx@ifpackageloaded{listings}{
911 \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
912 \newcommand\lstinputmhlisting[2][]{\%
913 \def\lst@mhrepos{}\setkeys{lst}{#1}%
914 \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
915 \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
916 }{}
917 }
918
919 </package>

```

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

Chapter 26

STEX -References Implementation

```
920 <*package>
921
922 %%%%%%%%%% references.dtx %%%%%%%%%%
923
924 <@@=stex_refs>
    Warnings and error messages
925
```

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

```
926 %\iow_new:N \c__stex_refs_refs_iow
927 \AtBeginDocument{
928 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
929 }
930 \AtEndDocument{
931 % \iow_close:N \c__stex_refs_refs_iow
932 }
```

`\STEXreftitle`

```
933 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
934
935 \NewDocumentCommand \STEXreftitle { m } {
936   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
937 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
938 \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:`

```
939 \cs_new_protected:Nn \stex_get_document_uri: {
940   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
941   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
942   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
943   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
944   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
945
946   \str_clear:N \l_tmpa_str
947   \prop_if_exist:NT \l_stex_current_repository_prop {
948     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
949       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
950     }
951   }
952
953   \str_if_empty:NTF \l_tmpa_str {
954     \str_set:Nx \l_stex_current_docns_str {
955       file:/\stex_path_to_string:N \l_tmpa_seq
956     }
957   }{
958     \bool_set_true:N \l_tmpa_bool
959     \bool_while_do:Nn \l_tmpa_bool {
960       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
961       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
962         {source} { \bool_set_false:N \l_tmpa_bool }
963       }{}{
964         \seq_if_empty:NT \l_tmpa_seq {
965           \bool_set_false:N \l_tmpa_bool
966         }
967       }
968     }
969
970     \seq_if_empty:NTF \l_tmpa_seq {
971       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
972     }{
973       \str_set:Nx \l_stex_current_docns_str {
974         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
975       }
976     }
977   }
978 }
```

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

`\l_stex_current_docurl_str`

```
979 \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:`

```
980 \cs_new_protected:Nn \stex_get_document_url: {
981   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
982   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
983   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

984 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
985 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
986
987 \str_clear:N \l_tmpa_str
988 \prop_if_exist:NT \l_stex_current_repository_prop {
989   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
990     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
991       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
992     }
993   }
994 }
995
996 \str_if_empty:NTF \l_tmpa_str {
997   \str_set:Nx \l_stex_current_docurl_str {
998     file:/\stex_path_to_string:N \l_tmpa_seq
999   }
1000 }{
1001   \bool_set_true:N \l_tmpa_bool
1002   \bool_while_do:Nn \l_tmpa_bool {
1003     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1004     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1005       {source} { \bool_set_false:N \l_tmpa_bool }
1006     }{}{
1007       \seq_if_empty:NT \l_tmpa_seq {
1008         \bool_set_false:N \l_tmpa_bool
1009       }
1010     }
1011   }
1012
1013   \seq_if_empty:NTF \l_tmpa_seq {
1014     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1015   }{
1016     \str_set:Nx \l_stex_current_docurl_str {
1017       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1018     }
1019   }
1020 }
1021 }

```

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

26.2 Setting Reference Targets

```

1022 \str_const:Nn \c__stex_refs_url_str{URL}
1023 \str_const:Nn \c__stex_refs_ref_str{REF}
1024 \str_new:N \l__stex_refs_curr_label_str
1025 % @currentlabel -> number
1026 % @currentlabelname -> title
1027 % @currentHref -> name.number <- id of some kind
1028 % \theH# -> \arabic{section}
1029 % \the# -> number
1030 % \hyper@makecurrent{#}
1031 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1032 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1033   \stex_get_document_uri:
1034   \str_clear:N \l__stex_refs_curr_label_str
1035   \str_set:Nx \l_tmpa_str { #1 }
1036   \str_if_empty:NT \l_tmpa_str {
1037     \int_incr:N \l__stex_refs_unnamed_counter_int
1038     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1039   }
1040   \str_set:Nx \l__stex_refs_curr_label_str {
1041     \l_stex_current_docns_str?\l_tmpa_str
1042   }
1043   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1044     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1045   }
1046   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1047     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1048   }
1049   \stex_if_smsmode:TF {
1050     \stex_get_document_url:
1051     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1052     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1053   }{
1054     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1055     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1056     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1057     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1058   }
1059 }

```

(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.

```

1060 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1061   \str_set:Nn \l_tmpa_str {#1?#2}
1062   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1063   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1064     \seq_new:c {g__stex_refs_labels_#2_seq}
1065   }
1066   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1067     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1068   }
1069 }

```

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

```

1070 \AtEndDocument{
1071   \def\stexauxadddocref#1 #2 {}{}
1072 }

```

`\stex_ref_new_sym_target:n`

```

1073 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1074   \stex_if_smsmode:TF {
1075     \str_if_exist:cF{sref_sym_#1_type}{
1076       \stex_get_document_url:
1077       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1078     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1079   }
1080 }{
1081   \str_if_empty:NF \l__stex_refs_curr_label_str {
1082     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1083     \immediate\write\@auxout{
1084       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1085         \l__stex_refs_curr_label_str
1086       }
1087     }
1088   }
1089 }
1090 }

```

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

26.3 Using References

```

1091 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1092
1093 \keys_define:nn { stex / sref } {
1094   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1095   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1096   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1097   post          .tl_set:N = \l__stex_refs_post_tl ,
1098 }
1099 \cs_new_protected:Nn \__stex_refs_args:n {
1100   \tl_clear:N \l__stex_refs_linktext_tl
1101   \tl_clear:N \l__stex_refs_fallback_tl
1102   \tl_clear:N \l__stex_refs_pre_tl
1103   \tl_clear:N \l__stex_refs_post_tl
1104   \str_clear:N \l__stex_refs_repo_str
1105   \keys_set:nn { stex / sref } { #1 }
1106 }

```

The actual macro:

```

1107 \NewDocumentCommand \sref { 0{} m}{
1108   \__stex_refs_args:n { #1 }
1109   \str_if_empty:NTF \l__stex_refs_indocument_str {
1110     \str_set:Nx \l_tmpa_str { #2 }
1111     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1112     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1113       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1114         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1115           \str_clear:N \l_tmpa_str
1116         }
1117       }{
1118         \str_clear:N \l_tmpa_str
1119       }
1120     }{
1121       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1122       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1123 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1124 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1125   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1126   \str_clear:N \l_tmpa_str
1127   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1128     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1129       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1130     }{
1131       \seq_map_break:n {
1132         \str_set:Nn \l_tmpa_str { ##1 }
1133       }
1134     }
1135   }
1136 }{
1137   \str_clear:N \l_tmpa_str
1138 }
1139 }
1140 \str_if_empty:NTF \l_tmpa_str {
1141   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1142 }{
1143   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1144     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1145       \cs_if_exist:cTF{autoref}{
1146         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1147       }{
1148         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1149       }
1150     }{
1151       \ltx@ifpackageloaded{hyperref}{
1152         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1153       }{
1154         \l__stex_refs_linktext_tl
1155       }
1156     }
1157   }{
1158     \ltx@ifpackageloaded{hyperref}{
1159       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1160     }{
1161       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1162     }
1163   }
1164 }
1165 }{
1166   % TODO
1167 }
1168 }

```

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

\srefsym

```

1169 \NewDocumentCommand \srefsym { 0{} m}{
1170   \stex_get_symbol:n { #2 }
1171   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1172 }

```

```

1173
1174 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1175   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1176     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1177   }{
1178     \__stex_refs_args:n { #1 }
1179     \str_if_empty:NTF \l__stex_refs_indocument_str {
1180       \tl_if_exist:cTF{sref_sym_#2 _type}{
1181         % doc uri in \l_tmpb_str
1182         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1183         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1184           % reference
1185           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1186             \cs_if_exist:cTF{autoref}{
1187               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1188             }{
1189               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1190             }
1191           }{
1192             \ltx@ifpackageloaded{hyperref}{
1193               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1194             }{
1195               \l__stex_refs_linktext_tl
1196             }
1197           }
1198         }{
1199           % URL
1200           \ltx@ifpackageloaded{hyperref}{
1201             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1202           }{
1203             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1204           }
1205         }
1206       }{
1207         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1208       }
1209     }{
1210       % TODO
1211     }
1212   }
1213 }

```

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

\srefsymuri

```

1214 \cs_new_protected:Npn \srefsymuri #1 #2 {
1215   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1216 }

```

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

```

1217 </package>

```

Chapter 27

STEX -Modules Implementation

```
1218 <*package>
1219
1220 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1221
1222 <@@=stex_modules>
1223
1224   Warnings and error messages
1225   \msg_new:nnn{stex}{error/unknownmodule}{
1226     No~module~#1~found
1227   }
1228   \msg_new:nnn{stex}{error/syntax}{
1229     Syntax~error:~#1
1230   }
1231   \msg_new:nnn{stex}{error/siglanguage}{
1232     Module~#1~declares~signature~#2,~but~does~not~
1233     declare~its~language
1234   }
1235   \msg_new:nnn{stex}{warning/deprecated}{
1236     #1~is~deprecated;~please~use~#2~instead!
1237   }
1238   \msg_new:nnn{stex}{error/conflictingmodules}{
1239     Conflicting~imports~for~module~#1
1240   }
```

`\l_stex_current_module_str` The current module:

```
1240 \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

```
1241 \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
1242 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1243   \str_if_empty:NTF \l_stex_current_module_str
1244   \prg_return_false: \prg_return_true:
1245 }

(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
1246 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1247   \prop_if_exist:cTF { c_stex_module_#1_prop }
1248   \prg_return_true: \prg_return_false:
1249 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1250 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1251   \stex_add_to_current_module:n { #1 }
1252   \stex_do_up_to_module:n { #1 }
1253 }}
1254 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1255
1256 \cs_new_protected:Nn \stex_add_to_current_module:n {
1257   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1258 }
1259 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1260 \cs_new_protected:Npn \STEXexport {
1261   \beginngroup
1262   \newlinechar=-1\relax
1263   \endlinechar=-1\relax
1264   %\catcode'\ = 9\relax
1265   \expandafter\endgroup\__stex_modules_export:n
1266 }
1267 \cs_new_protected:Nn \__stex_modules_export:n {
1268   \ignorespaces #1
1269   \stex_add_to_current_module:n { \ignorespaces #1 }
1270   \stex_smsmode_do:
1271 }
1272 \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
1273 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1274   \str_set:Nx \l_tmpa_str { #1 }
1275   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1276 }

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

```

`\stex_add_import_to_current_module:n`

```

1277 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1278   \str_set:Nx \l_tmpa_str { #1 }
1279   \exp_args:Nno
1280   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1281     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1282   }
1283 }

```

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

`\stex_collect_imports:n`

```

1284 \cs_new_protected:Nn \stex_collect_imports:n {
1285   \seq_clear:N \l_stex_collect_imports_seq
1286   \__stex_modules_collect_imports:n {#1}
1287 }
1288 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1289   \seq_map_inline:cn {c_stex_module_#1_imports} {
1290     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1291       \__stex_modules_collect_imports:n { ##1 }
1292     }
1293   }
1294   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1295     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1296   }
1297 }

```

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

`\stex_do_up_to_module:n`

```

1298 \int_new:N \l__stex_modules_group_depth_int
1299 \cs_new_protected:Nn \stex_do_up_to_module:n {
1300   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1301     #1
1302   }{
1303     #1
1304     \expandafter \tl_gset:Nn
1305     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1306     \expandafter\expandafter\expandafter\endcsname
1307     \expandafter\expandafter\expandafter { \csname
1308       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1309     \aftergroup\__stex_modules_aftergroup_do:
1310   }
1311 }
1312 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1313 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1314   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1315     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1316   }}
1317   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1318     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1319     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1320   }{
1321     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1322 \aftergroup\__stex_modules_aftergroup_do:
1323 }
1324 }
1325 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1326 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1327 }

```

(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).

```

1328

```

(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.

```

1329 \str_new:N \l_stex_module_ns_str
1330 \str_new:N \l_stex_module_subpath_str
1331 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1332 \seq_set_eq:NN \l_tmpa_seq #2
1333 % split off file extension
1334 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1335 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1336 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1337 \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1338
1339 \bool_set_true:N \l_tmpa_bool
1340 \bool_while_do:Nn \l_tmpa_bool {
1341 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1342 \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1343 {source} { \bool_set_false:N \l_tmpa_bool }
1344 }{}{
1345 \seq_if_empty:NT \l_tmpa_seq {
1346 \bool_set_false:N \l_tmpa_bool
1347 }
1348 }
1349 }
1350
1351 \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1352 % \l_tmpa_seq <- sub-path relative to archive
1353 \str_if_empty:NTF \l_stex_module_subpath_str {
1354 \str_set:Nx \l_stex_module_ns_str {#1}
1355 }{
1356 \str_set:Nx \l_stex_module_ns_str {
1357 #1/\l_stex_module_subpath_str
1358 }
1359 }
1360 }
1361
1362 \cs_new_protected:Nn \stex_modules_current_namespace: {
1363 \str_clear:N \l_stex_module_subpath_str
1364 \prop_if_exist:NTF \l_stex_current_repository_prop {
1365 \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1366     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1367   }{
1368     % split off file extension
1369     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1370     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1371     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1372     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1373     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1374     \str_set:Nx \l_stex_module_ns_str {
1375       file:/\stex_path_to_string:N \l_tmpa_seq
1376     }
1377   }
1378 }

```

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

27.1 The smodule environment

smodule arguments:

```

1379 \keys_define:nn { stex / module } {
1380   title      .tl_set:N      = \smodulename ,
1381   type       .str_set_x:N   = \smodulename ,
1382   id         .str_set_x:N   = \smoduleid ,
1383   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1384   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1385   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1386   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1387   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1388   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1389   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1390   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1391 }
1392
1393 \cs_new_protected:Nn \__stex_modules_args:n {
1394   \str_clear:N \smodulename
1395   \str_clear:N \smodulename
1396   \str_clear:N \smoduleid
1397   \str_clear:N \l_stex_module_ns_str
1398   \str_clear:N \l_stex_module_deprecate_str
1399   \str_clear:N \l_stex_module_lang_str
1400   \str_clear:N \l_stex_module_sig_str
1401   \str_clear:N \l_stex_module_creators_str
1402   \str_clear:N \l_stex_module_contributors_str
1403   \str_clear:N \l_stex_module_meta_str
1404   \str_clear:N \l_stex_module_srccite_str
1405   \keys_set:nn { stex / module } { #1 }
1406 }
1407
1408 % module parameters here? In the body?
1409

```

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

```

1410 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1411 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1412 \str_set:Nx \l_stex_module_name_str { #2 }
1413 \__stex_modules_args:n { #1 }

```

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

```

1414 \stex_if_in_module:TF {
1415   % Nested module
1416   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1417   { ns } \l_stex_module_ns_str
1418   \str_set:Nx \l_stex_module_name_str {
1419     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1420     { name } / \l_stex_module_name_str
1421   }
1422   \str_if_empty:NT \l_stex_module_lang_str {
1423     \str_set:Nx \l_stex_module_lang_str {
1424       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1425       { lang }
1426     }
1427   }
1428 }{
1429   % not nested:
1430   \str_if_empty:NT \l_stex_module_ns_str {
1431     \stex_modules_current_namespace:
1432     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1433       / {\l_stex_module_ns_str}
1434     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1435     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1436       \str_set:Nx \l_stex_module_ns_str {
1437         \stex_path_to_string:N \l_tmpa_seq
1438       }
1439     }
1440   }
1441 }

```

Next, we determine the language of the module:

```

1442 \str_if_empty:NT \l_stex_module_lang_str {
1443   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1444   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1445   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1446   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1447     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1448       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1449     }
1450   }
1451   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1452   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1453     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1454     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1455       inferred~from~file~name}
1456   }
1457 }
1458
1459 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1460 \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1461 }}

```

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

```

1462 \str_if_empty:NTF \l_stex_module_sig_str {
1463   \exp_args:Nnx \prop_gset_from_keyval:cn {
1464     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1465   } {
1466     name      = \l_stex_module_name_str ,
1467     ns        = \l_stex_module_ns_str ,
1468     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1469     lang      = \l_stex_module_lang_str ,
1470     sig       = \l_stex_module_sig_str ,
1471     deprecate = \l_stex_module_deprecate_str ,
1472     meta      = \l_stex_module_meta_str
1473   }
1474   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1475   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1476   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1477   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1478   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1479 \str_if_empty:NT \l_stex_module_meta_str {
1480   \str_set:Nx \l_stex_module_meta_str {
1481     \c_stex_metatheory_ns_str ? Metatheory
1482   }
1483 }
1484 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1485   \bool_set_true:N \l_stex_in_meta_bool
1486   \exp_args:Nx \stex_add_to_current_module:n {
1487     \bool_set_true:N \l_stex_in_meta_bool
1488     \stex_activate_module:n {\l_stex_module_meta_str}
1489     \bool_set_false:N \l_stex_in_meta_bool
1490   }
1491   \stex_activate_module:n {\l_stex_module_meta_str}
1492   \bool_set_false:N \l_stex_in_meta_bool
1493 }
1494 }{
1495   \str_if_empty:NT \l_stex_module_lang_str {
1496     \msg_error:nnxx{stex}{error/siglanguage}{
1497       \l_stex_module_ns_str?\l_stex_module_name_str
1498     }\l_stex_module_sig_str}
1499   }
1500   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1501   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1502     \stex_debug:nn{modules}{(already exists)}
1503   }{
1504     \stex_debug:nn{modules}{(needs loading)}
1505     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1506     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1507     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1508     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1509     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1510     \str_set:Nx \l_tmpa_str {
1511       \stex_path_to_string:N \l_tmpa_seq /
1512       \l_tmpa_str . \l_stex_module_sig_str .tex
1513     }
1514     \IfFileExists \l_tmpa_str {
1515       \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1516         \str_clear:N \l_stex_current_module_str
1517         \seq_clear:N \l_stex_all_modules_seq
1518         \stex_debug:nn{modules}{Loading~signature}
1519       }
1520     }{
1521       \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1522     }
1523   }
1524   \stex_if_smsmode:F {
1525     \stex_activate_module:n {
1526       \l_stex_module_ns_str ? \l_stex_module_name_str
1527     }
1528   }
1529   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1530 }
1531 \str_if_empty:NF \l_stex_module_deprecate_str {
1532   \msg_warning:nnxx{stex}{warning/deprecated}{
1533     Module~\l_stex_current_module_str
1534   }{
1535     \l_stex_module_deprecate_str
1536   }
1537 }
1538 \seq_put_right:Nx \l_stex_all_modules_seq {
1539   \l_stex_module_ns_str ? \l_stex_module_name_str
1540 }
1541 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1542 }

```

(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}
1543 \cs_new_protected:Nn \__stex_modules_begin_module: {
1544   \stex_reactivate_macro:N \STEXexport
1545   \stex_reactivate_macro:N \importmodule
1546   \stex_reactivate_macro:N \symdecl
1547   \stex_reactivate_macro:N \notation
1548   \stex_reactivate_macro:N \symdef
1549
1550   \stex_debug:nn{modules}{
1551     New~module:\\
1552     Namespace:~\l_stex_module_ns_str\\
1553     Name:~\l_stex_module_name_str\\
1554     Language:~\l_stex_module_lang_str\\
1555     Signature:~\l_stex_module_sig_str\\
1556     Metatheory:~\l_stex_module_meta_str\\

```

```

1557   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1558 }
1559
1560 \stex_if_do_html:T{
1561   \begin{stex_annotate_env} {theory} {
1562     \l_stex_module_ns_str ? \l_stex_module_name_str
1563   }
1564
1565   \stex_annotate_invisible:nnn{header}{} {
1566     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1567     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1568     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1569       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1570     }
1571     \str_if_empty:NF \smoduletype {
1572       \stex_annotate:nnn{type}{\smoduletype}{}
1573     }
1574   }
1575 }
1576 % TODO: Inherit metatheory for nested modules?
1577 }
1578 \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}

```

1579 \cs_new_protected:Nn \_stex_modules_end_module: {
1580   \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1581   \stex_reset_up_to_module:n \l_stex_current_module_str
1582   \stex_if_smsmode:T {
1583     \stex_persist:x {
1584       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1585         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1586       }
1587       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1588         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1589       }
1590       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1591         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1592       }
1593       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1594     }
1595     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1596     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1597   }
1598 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1599 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1600 \NewDocumentEnvironment { smodule } { 0 } { m } {
1601   \stex_module_setup:nn{#1}{#2}
1602   %\par
1603   \stex_if_smsmode:F{

```



```

1604 \tl_if_empty:NF \smodulename {
1605   \exp_args:No \stex_document_title:n \smodulename
1606 }
1607 \tl_clear:N \l_tmpa_tl
1608 \clist_map_inline:Nn \smodulename {
1609   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1610     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1611   }
1612 }
1613 \tl_if_empty:NTF \l_tmpa_tl {
1614   \__stex_modules_smodule_start:
1615 }{
1616   \l_tmpa_tl
1617 }
1618 }
1619 \__stex_modules_begin_module:
1620 \str_if_empty:NF \smoduleid {
1621   \stex_ref_new_doc_target:n \smoduleid
1622 }
1623 \stex_smsmode_do:
1624 } {
1625   \__stex_modules_end_module:
1626   \stex_if_smsmode:F {
1627     \end{stex_annotate_env}
1628     \clist_set:Nn \l_tmpa_clist \smodulename
1629     \tl_clear:N \l_tmpa_tl
1630     \clist_map_inline:Nn \l_tmpa_clist {
1631       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1632         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1633       }
1634     }
1635     \tl_if_empty:NTF \l_tmpa_tl {
1636       \__stex_modules_smodule_end:
1637     }{
1638       \l_tmpa_tl
1639     }
1640   }
1641 }

```

\stexpatchmodule

```

1642 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1643 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1644
1645 \newcommand\stexpatchmodule[3] [] {
1646   \str_set:Nx \l_tmpa_str{ #1 }
1647   \str_if_empty:NTF \l_tmpa_str {
1648     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1649     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1650   }{
1651     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1652     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1653   }
1654 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1655 \NewDocumentCommand \STEXModule { m } {
1656   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1657   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1658   \tl_set:Nn \l_tmpa_tl {
1659     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1660   }
1661   \seq_map_inline:Nn \l_stex_all_modules_seq {
1662     \str_set:Nn \l_tmpb_str { ##1 }
1663     \str_if_eq:eeT { \l_tmpa_str } {
1664       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1665     } {
1666       \seq_map_break:n {
1667         \tl_set:Nn \l_tmpa_tl {
1668           \stex_invoke_module:n { ##1 }
1669         }
1670       }
1671     }
1672   }
1673   \l_tmpa_tl
1674 }
1675
1676 \cs_new_protected:Nn \stex_invoke_module:n {
1677   \stex_debug:nn{modules}{Invoking~module~#1}
1678   \peek_charcode_remove:NTF ! {
1679     \__stex_modules_invoke_uri:nN { #1 }
1680   } {
1681     \peek_charcode_remove:NTF ? {
1682       \__stex_modules_invoke_symbol:nn { #1 }
1683     } {
1684       \msg_error:nnx{stex}{error/syntax}{
1685         ?~or~!~expected~after~
1686         \c_backslash_str STEXModule{#1}
1687       }
1688     }
1689   }
1690 }
1691
1692 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1693   \str_set:Nn #2 { #1 }
1694 }
1695
1696 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1697   \stex_invoke_symbol:n{#1?#2}
1698 }

```

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

```

\stex_activate_module:n
1699 \bool_new:N \l_stex_in_meta_bool
1700 \bool_set_false:N \l_stex_in_meta_bool

```

```

1701 \cs_new_protected:Nn \stex_activate_module:n {
1702   \stex_debug:nn{modules}{Activating~module~#1}
1703   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1704     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1705     \use:c{ c_stex_module_#1_code }
1706   }
1707 }

```

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

```

1708 \endpackage

```

Chapter 28

STEX -Module Inheritance Implementation

```
1709 <*package>
1710
1711 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1712
```

28.1 SMS Mode

```
1713 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1714 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1715 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1716 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1717
1718 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1719   \makeatletter
1720   \makeatother
1721   \ExplSyntaxOn
1722   \ExplSyntaxOff
1723   \rustexBREAK
1724 }
1725
1726 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1727   \symdef
1728   \importmodule
1729   \notation
1730   \symdecl
1731   \STEXexport
1732   \inlineass
1733   \inlinedef
1734   \inlineex
1735   \endinput
1736   \setnotation
```

```

1737 \copynotation
1738 \assign
1739 \renamedekl
1740 \donotcopy
1741 \instantiate
1742 }
1743
1744 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1745   \tl_to_str:n {
1746     smodule,
1747     copymodule,
1748     interpretmodule,
1749     sdefinition,
1750     sexample,
1751     sassertion,
1752     sparagraph,
1753     mathstructure
1754   }
1755 }

```

(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
1756 \bool_new:N \g__stex_smsmode_bool
1757 \bool_set_false:N \g__stex_smsmode_bool
1758 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1759   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1760 }

```

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

```

\_stex_smsmode_in_smsmode:nn
1761 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1762   \vbox_set:Nn \l_tmpa_box {
1763     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1764     \bool_gset_true:N \g__stex_smsmode_bool
1765     #2
1766     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1767   }
1768   \box_clear:N \l_tmpa_box
1769 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

```

\stex_file_in_smsmode:nn
1770 \quark_new:N \q__stex_smsmode_break
1771
1772 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1773   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1774   \stex_smsmode_do:
1775 }
1776
1777 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1778   \_stex_modules_args:n{#1}

```

```

1779 \stex_if_in_module:F {
1780   \str_if_empty:NF \l_stex_module_sig_str {
1781     \stex_modules_current_namespace:
1782     \str_set:Nx \l_stex_module_name_str { #2 }
1783     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1784       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1785       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1786       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1787       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1788       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1789       \str_set:Nx \l_tmpa_str {
1790         \stex_path_to_string:N \l_tmpa_seq /
1791         \l_tmpa_str . \l_stex_module_sig_str .tex
1792       }
1793       \IfFileExists \l_tmpa_str {
1794         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1795       }{
1796         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1797       }
1798     }
1799   }
1800 }
1801 }
1802
1803 \prg_new_conditional:Nnn \__stex_smsmode_check_import_pair:nn {T,F,TF} {
1804   %\stex_debug:nn{import-pair}{\detokenize{#{1}~#{2}}}}
1805   \tl_if_empty:nTF{#{1}}{
1806     \prop_if_exist:NTF \l_stex_current_repository_prop
1807     {
1808       %\stex_debug:nn{import-pair}{in repository \prop_item:Nn \l_stex_current_repository_
1809       \prg_return_true:
1810     } {
1811       \seq_set_split:Nnn \l_tmpa_seq ? {#2}
1812       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
1813       \tl_if_empty:NT \l_tmpa_tl {
1814         \seq_pop_left:NN \l_tmpa_seq \l_tmpa_tl
1815       }
1816       %\stex_debug:nn{import-pair}{\seq_use:Nn \l_tmpa_seq,~of~length~\seq_count:N \l_tmpa
1817       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1
1818         \prg_return_true: \prg_return_false:
1819     }
1820   }\prg_return_true:
1821 }
1822
1823 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1824   \stex_filestack_push:n{#1}
1825   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1826   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1827   % ----- new -----
1828   \__stex_smsmode_in_smsmode:nn{#1}{
1829     \let\importmodule\__stex_smsmode_importmodule:
1830     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1831     \let\__stex_modules_begin_module:\relax
1832     \let\__stex_modules_end_module:\relax

```

```

1833 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1834 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1835 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1836 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1837 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1838 \everyeof{\q__stex_smsmode_break\noexpand}
1839 \expandafter\expandafter\expandafter
1840 \stex_smsmode_do:
1841 \csname @ @ input\endcsname "#1"\relax
1842
1843 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1844   \stex_filestack_push:n{##1}
1845   \expandafter\expandafter\expandafter
1846   \stex_smsmode_do:
1847   \csname @ @ input\endcsname "##1"\relax
1848   \stex_filestack_pop:
1849 }
1850 }
1851 % ----- new -----
1852 \__stex_smsmode_in_smsmode:nn{#1} {
1853   #2
1854   % ----- new -----
1855   \begingroup
1856   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1857   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1858     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1859       \stex_import_module_uri:nn ##1
1860       \stex_import_require_module:nnnn
1861       \l_stex_import_ns_str
1862       \l_stex_import_archive_str
1863       \l_stex_import_path_str
1864       \l_stex_import_name_str \endgroup
1865     }
1866   }
1867   \endgroup
1868   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1869   % ----- new -----
1870   \everyeof{\q__stex_smsmode_break\noexpand}
1871   \expandafter\expandafter\expandafter
1872   \stex_smsmode_do:
1873   \csname @ @ input\endcsname "#1"\relax
1874 }
1875 \stex_filestack_pop:
1876 }

```

(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:

```

1877 \cs_new_protected:Npn \stex_smsmode_do: {
1878   \stex_if_smsmode:T {
1879     \__stex_smsmode_do:w
1880   }
1881 }

```

```

1882 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1883   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1884     \expandafter\if\expandafter\relax\noexpand#1
1885     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1886     \else\expandafter\__stex_smsmode_do:w\fi
1887   }{
1888     \__stex_smsmode_do:w %#1
1889   }
1890 }
1891 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1892   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1893     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1894       #1\__stex_smsmode_do:w
1895     }{
1896       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1897         #1
1898       }{
1899         \cs_if_eq:NNTF \begin #1 {
1900           \__stex_smsmode_check_begin:n
1901         }{
1902           \cs_if_eq:NNTF \end #1 {
1903             \__stex_smsmode_check_end:n
1904           }{
1905             \__stex_smsmode_do:w
1906           }
1907         }
1908       }
1909     }
1910   }
1911 }
1912
1913 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1914   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1915     \begin{#1}
1916   }{
1917     \__stex_smsmode_do:w
1918   }
1919 }
1920 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1921   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1922     \end{#1}\__stex_smsmode_do:w
1923   }{
1924     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1925   }
1926 }

```

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

28.2 Inheritance

```

1927 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1928 \cs_new_protected:Nn \stex_import_module_uri:nn {

```



```

1929 \str_set:Nx \l_stex_import_archive_str { #1 }
1930 \str_set:Nn \l_stex_import_path_str { #2 }
1931
1932 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1933 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1934 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1935
1936 \stex_modules_current_namespace:
1937 \bool_lazy_all:nTF {
1938   {\str_if_empty_p:N \l_stex_import_archive_str}
1939   {\str_if_empty_p:N \l_stex_import_path_str}
1940   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1941 }{
1942   \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1943   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1944 }{
1945   \str_if_empty:NT \l_stex_import_archive_str {
1946     \prop_if_exist:NT \l_stex_current_repository_prop {
1947       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1948     }
1949   }
1950   \str_if_empty:NTF \l_stex_import_archive_str {
1951     \str_if_empty:NF \l_stex_import_path_str {
1952       \stex_path_from_string:Nn \l_tmpb_seq {
1953         \l_stex_module_ns_str / .. / \l_stex_import_path_str
1954       }
1955       \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1956       \str_replace_once:Nnn \l_stex_import_ns_str {file:/{ } {file://}
1957     }
1958   }{
1959     \stex_require_repository:n \l_stex_import_archive_str
1960     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str_manifest_prop } { ns }
1961     \l_stex_import_ns_str
1962     \str_if_empty:NF \l_stex_import_path_str {
1963       \str_set:Nx \l_stex_import_ns_str {
1964         \l_stex_import_ns_str / \l_stex_import_path_str
1965       }
1966     }
1967   }
1968 }
1969 }

```

(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 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str \str_new:N \l_stex_import_path_str
\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>}
1974 \cs_new_protected:Nn \stex_import_require_module:nnnn {

```

```

1975 \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1976
1977   \stex_debug:nn{requiremodule}{Here:\~1:~#1\~2:~#2\~3:~#3\~4:~#4}
1978
1979   \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1980   \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1981
1982   %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
1983
1984   % archive
1985   \str_set:Nx \l_tmpa_str { #2 }
1986   \str_if_empty:NTF \l_tmpa_str {
1987     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1988     \seq_put_right:Nn \l_tmpa_seq {...}
1989   } {
1990     \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1991     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1992     \seq_put_right:Nn \l_tmpa_seq { source }
1993   }
1994
1995   % path
1996   \str_set:Nx \l_tmpb_str { #3 }
1997   \str_if_empty:NTF \l_tmpb_str {
1998     \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1999
2000     \ltx@ifpackageloaded{babel} {
2001       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2002         { \language } \l_tmpb_str {
2003         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2004       }
2005     } {
2006       \str_clear:N \l_tmpb_str
2007     }
2008
2009     \stex_debug:nn{modules}{Checking~a1~\l_tmpa_str.\l_tmpb_str.tex}
2010     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2011       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2012     }{
2013       \stex_debug:nn{modules}{Checking~a2~\l_tmpa_str.tex}
2014       \IfFileExists{ \l_tmpa_str.tex }{
2015         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2016       }{
2017         % try english as default
2018         \stex_debug:nn{modules}{Checking~a3~\l_tmpa_str.en.tex}
2019         \IfFileExists{ \l_tmpa_str.en.tex }{
2020           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2021         }{
2022           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2023         }
2024       }
2025     }
2026
2027   } {
2028     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str

```

```

2029 \seq_concat:NNN \l_tmpb_seq \l_tmpa_seq \l_tmpb_seq
2030
2031 \ltx@ifpackageloaded{babel} {
2032   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
2033     { \language } \l_tmpb_str {
2034       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
2035     }
2036 } {
2037   \str_clear:N \l_tmpb_str
2038 }
2039
2040 \stex_path_canonicalize:N \l_tmpb_seq
2041 \stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
2042
2043 \stex_debug:nn{modules}{Checking~b1~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
2044 \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
2045   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
2046 }{
2047   \stex_debug:nn{modules}{Checking~b2~\l_tmpa_str/\l_tmpc_str.tex}
2048   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
2049     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
2050   }{
2051     % try english as default
2052     \stex_debug:nn{modules}{Checking~b3~\l_tmpa_str/\l_tmpc_str.en.tex}
2053     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
2054       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
2055     }{
2056       \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.\l_tmpb_str.tex}
2057       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2058         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2059       }{
2060         \stex_debug:nn{modules}{Checking~b4~\l_tmpa_str.tex}
2061         \IfFileExists{ \l_tmpa_str.tex }{
2062           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2063         }{
2064           % try english as default
2065           \stex_debug:nn{modules}{Checking~b5~\l_tmpa_str.en.tex}
2066           \IfFileExists{ \l_tmpa_str.en.tex }{
2067             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2068           }{
2069             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2070           }
2071         }
2072       }
2073     }
2074   }
2075 }
2076
2077
2078 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2079   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2080     \seq_clear:N \l_stex_all_modules_seq
2081     \str_clear:N \l_stex_current_module_str
2082     \str_set:Nx \l_tmpb_str { #2 }

```

```

2083     \str_if_empty:NF \l_tmpb_str {
2084       \stex_set_current_repository:n { #2 }
2085     }
2086     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2087   }
2088
2089   \stex_if_module_exists:nF { #1 ? #4 } {
2090     \msg_error:nnx{stex}{error/unknownmodule}{
2091       #1?#4~(in~file~\g__stex_importmodule_file_str)
2092     }
2093   }
2094 }
2095
2096 }
2097 \stex_activate_module:n { #1 ? #4 }
2098 }

```

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

`\importmodule`

```

2099 \NewDocumentCommand \importmodule { 0{} m } {
2100   \stex_import_module_uri:nn { #1 } { #2 }
2101   \stex_debug:nn{modules}{Importing~module:~
2102     \l_stex_import_ns_str ? \l_stex_import_name_str
2103   }
2104   \stex_import_require_module:nnnn
2105   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2106   { \l_stex_import_path_str } { \l_stex_import_name_str }
2107   \stex_if_smsmode:F {
2108     \stex_annotate_invisible:nnn
2109     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2110   }
2111   \exp_args:Nx \stex_add_to_current_module:n {
2112     \stex_import_require_module:nnnn
2113     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2114     { \l_stex_import_path_str } { \l_stex_import_name_str }
2115   }
2116   \exp_args:Nx \stex_add_import_to_current_module:n {
2117     \l_stex_import_ns_str ? \l_stex_import_name_str
2118   }
2119   \stex_smsmode_do:
2120   \ignorespacesandpars
2121 }
2122 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2123 \NewDocumentCommand \usemodule { 0{} m } {
2124   \stex_if_smsmode:F {
2125     \stex_import_module_uri:nn { #1 } { #2 }
2126     \stex_import_require_module:nnnn
2127     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2128     { \l_stex_import_path_str } { \l_stex_import_name_str }
2129     \stex_annotate_invisible:nnn

```

```

2130     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2131   }
2132   \stex_smsmode_do:
2133   \ignorespacesandpars
2134 }

```

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

```

2135 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2136   \tl_if_empty:nF{#2}{
2137     \clist_set:Nn \l_tmpa_clist {#2}
2138     \clist_map_inline:Nn \l_tmpa_clist {
2139       \tl_if_head_eq_charcode:nNTF {##1}[{
2140         #1 ##1
2141       }{
2142         #1{##1}
2143       }
2144     }
2145   }
2146 }
2147 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2148
2149
2150 \</package>

```

Chapter 29

STEX -Symbols Implementation

```
2151 <*package>
2152
2153 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2154
2155 Warnings and error messages
2156 \msg_new:nnn{stex}{error/wrongargs}{
2157   args~value~in~symbol~declaration~for~#1~
2158   needs~to~be~i,~a,~b~or~B,~but~#2~given
2159 }
2160 \msg_new:nnn{stex}{error/unknownsymbol}{
2161   No~symbol~#1~found!
2162 }
2163 \msg_new:nnn{stex}{error/seqlength}{
2164   Expected~#1~arguments;~got~#2!
2165 }
2166 \msg_new:nnn{stex}{error/unknownnotation}{
2167   Unknown~notation~#1~for~#2!
2168 }
```

29.1 Symbol Declarations

```
2168 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2169 \cs_new_protected:Nn \stex_all_symbols:n {
2170   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2171   \seq_map_inline:Nn \l_stex_all_modules_seq {
2172     \seq_map_inline:cn{c_stex_module_##1_constants}{
2173       \__stex_symdecl_all_symbols_cs{##1?####1}
2174     }
2175   }
2176 }
```

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

\STEXsymbol

```
2177 \NewDocumentCommand \STEXsymbol { m } {  
2178   \stex_get_symbol:n { #1 }  
2179   \exp_args:No  
2180   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2181 }
```

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

symdecl arguments:

```
2182 \keys_define:nn { stex / symdecl } {  
2183   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2184   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2185   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2186   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2187   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2188   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2189   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2190   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2191   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2192   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2193   assoc     .choices:nn =  
2194     {bin,binl,binr,pre,conj,pwconj}  
2195     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2196 }  
2197  
2198 \bool_new:N \l_stex_symdecl_make_macro_bool  
2199  
2200 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2201   \str_clear:N \l_stex_symdecl_name_str  
2202   \str_clear:N \l_stex_symdecl_args_str  
2203   \str_clear:N \l_stex_symdecl_deprecate_str  
2204   \str_clear:N \l_stex_symdecl_reorder_str  
2205   \str_clear:N \l_stex_symdecl_assoctype_str  
2206   \bool_set_false:N \l_stex_symdecl_local_bool  
2207   \tl_clear:N \l_stex_symdecl_type_tl  
2208   \tl_clear:N \l_stex_symdecl_definiens_tl  
2209  
2210   \keys_set:nn { stex / symdecl } { #1 }  
2211 }
```

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

```
2212  
2213 \NewDocumentCommand \symdecl { s m O{} } {  
2214   \__stex_symdecl_args:n { #3 }  
2215   \IfBooleanTF #1 {  
2216     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2217   } {  
2218     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2219   }  
2220   \stex_symdecl_do:n { #2 }  
2221   \stex_smsmode_do:  
2222 }
```

```

2223
2224 \cs_new_protected:Nn \stex_symdecl_do:nn {
2225   \__stex_symdecl_args:n{#1}
2226   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2227   \stex_symdecl_do:n{#2}
2228 }
2229
2230 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2231 \cs_new_protected:Nn \stex_symdecl_do:n {
2232   \stex_if_in_module:F {
2233     % TODO throw error? some default namespace?
2234   }
2235
2236   \str_if_empty:NT \l_stex_symdecl_name_str {
2237     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2238   }
2239
2240   \prop_if_exist:cT { l_stex_symdecl_
2241     \l_stex_current_module_str ?
2242     \l_stex_symdecl_name_str
2243     _prop
2244   }{
2245     % TODO throw error (beware of circular dependencies)
2246   }
2247
2248   \prop_clear:N \l_tmpa_prop
2249   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2250   \seq_clear:N \l_tmpa_seq
2251   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2252   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2253
2254   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2255     \str_if_empty:NF \l_stex_module_deprecate_str {
2256       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2257     }
2258   }
2259   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2260
2261   \exp_args:No \stex_add_constant_to_current_module:n {
2262     \l_stex_symdecl_name_str
2263   }
2264
2265   % arity/args
2266   \int_zero:N \l_tmpb_int
2267
2268   \bool_set_true:N \l_tmpa_bool
2269   \str_map_inline:Nn \l_stex_symdecl_args_str {
2270     \token_case_meaning:NnF ##1 {
2271       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2272       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2273     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2274     {\tl_to_str:n a} {
2275         \bool_set_false:N \l_tmpa_bool
2276         \int_incr:N \l_tmpb_int
2277     }
2278     {\tl_to_str:n B} {
2279         \bool_set_false:N \l_tmpa_bool
2280         \int_incr:N \l_tmpb_int
2281     }
2282 }{
2283     \msg_error:nnxx{stex}{error/wrongargs}{
2284         \l_stex_current_module_str ?
2285         \l_stex_symdecl_name_str
2286     }{##1}
2287 }
2288 }
2289 \bool_if:NTF \l_tmpa_bool {
2290     % possibly numeric
2291     \str_if_empty:NTF \l_stex_symdecl_args_str {
2292         \prop_put:Nnn \l_tmpa_prop { args } {}
2293         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2294     }{
2295         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2296         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2297         \str_clear:N \l_tmpa_str
2298         \int_step_inline:nn \l_tmpa_int {
2299             \str_put_right:Nn \l_tmpa_str i
2300         }
2301         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2302     }
2303 } {
2304     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2305     \prop_put:Nnx \l_tmpa_prop { arity }
2306     { \str_count:N \l_stex_symdecl_args_str }
2307 }
2308 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2309
2310 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2311     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2312 }{
2313     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2314 }
2315
2316 % semantic macro
2317
2318 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2319     \exp_args:Nx \stex_do_up_to_module:n {
2320         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2321             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2322         }}
2323     }
2324 }
2325
2326 \stex_debug:nn{symbols}{New~symbol:~

```

```

2327 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2328 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2329 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2330 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2331 }
2332
2333 % circular dependencies require this:
2334 \stex_if_do_html:T {
2335   \stex_annotate_invisible:nnn {symdecl} {
2336     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2337   } {
2338     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2339       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2340     }
2341     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$}
2342     \prop_item:Nn \l_tmpa_prop { args }
2343   }
2344   \stex_annotate_invisible:nnn{macroname}{#1}{}
2345   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2346     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2347   }
2348   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2349     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2350   }
2351   \str_if_empty:NF \l_stex_symdecl_reorder_str {
2352     \stex_annotate_invisible:nnn{reorder_args}{\l_stex_symdecl_reorder_str}{}
2353   }
2354 }
2355 }
2356 }
2357 \prop_if_exist:cF {
2358   \l_stex_symdecl_
2359   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2360   _prop
2361 } {
2362   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2363     \__stex_symdecl_restore_symbol:nnnnnnn
2364       {\l_stex_symdecl_name_str}
2365       { \prop_item:Nn \l_tmpa_prop {args} }
2366       { \prop_item:Nn \l_tmpa_prop {arity} }
2367       { \prop_item:Nn \l_tmpa_prop {assoc} }
2368       { \prop_item:Nn \l_tmpa_prop {defined} }
2369       {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2370       {\l_stex_current_module_str}
2371 }
2372 }
2373 }
2374 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2375   \prop_clear:N \l_tmpa_prop
2376   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2377   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2378   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2379   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2380   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }

```

```

2381 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2382 \tl_if_empty:nF{#6}{
2383   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2384 }
2385 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2386 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2387 }

```

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

`\stex_get_symbol:n`

```

2388 \str_new:N \l_stex_get_symbol_uri_str
2389
2390 \cs_new_protected:Nn \stex_get_symbol:n {
2391   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2392     \tl_set:Nn \l_tmpa_tl { #1 }
2393     \__stex_symdecl_get_symbol_from_cs:
2394   }{
2395     % argument is a string
2396     % is it a command name?
2397     \cs_if_exist:cTF { #1 }{
2398       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2399       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2400       \str_if_empty:NTF \l_tmpa_str {
2401         \exp_args:Nx \cs_if_eq:NNTF {
2402           \tl_head:N \l_tmpa_tl
2403         } \stex_invoke_symbol:n {
2404           \__stex_symdecl_get_symbol_from_cs:
2405         }{
2406           \__stex_symdecl_get_symbol_from_string:n { #1 }
2407         }
2408       } {
2409         \__stex_symdecl_get_symbol_from_string:n { #1 }
2410       }
2411     }{
2412       % argument is not a command name
2413       \__stex_symdecl_get_symbol_from_string:n { #1 }
2414       % \l_stex_all_symbols_seq
2415     }
2416   }
2417   \str_if_eq:eeF {
2418     \prop_item:cn {
2419       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2420     }{ deprecate }
2421   }{
2422     \msg_warning:nxxx{stex}{warning/deprecated}{
2423       Symbol~\l_stex_get_symbol_uri_str
2424     }{
2425       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2426     }
2427   }
2428 }
2429
2430 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2431 \tl_set:Nn \l_tmpa_tl {
2432   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2433 }
2434 \str_set:Nn \l_tmpa_str { #1 }
2435
2436 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2437
2438 \str_if_in:NnTF \l_tmpa_str ? {
2439   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2440   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2441   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2442 }{
2443   \str_clear:N \l_tmpb_str
2444 }
2445 \str_if_empty:NnTF \l_tmpb_str {
2446   \seq_map_inline:Nn \l_stex_all_modules_seq {
2447     \seq_map_inline:cn{c_stex_module_###1_constants}{
2448       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2449         \seq_map_break:n{\seq_map_break:n{
2450           \tl_set:Nn \l_tmpa_tl {
2451             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2452           }
2453         }}
2454       }
2455     }
2456   }
2457 }{
2458   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2459   \seq_map_inline:Nn \l_stex_all_modules_seq {
2460     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2461       \seq_map_inline:cn{c_stex_module_###1_constants}{
2462         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2463           \seq_map_break:n{\seq_map_break:n{
2464             \tl_set:Nn \l_tmpa_tl {
2465               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2466             }
2467           }}
2468         }
2469       }
2470     }
2471   }
2472 }
2473
2474 \l_tmpa_tl
2475 }
2476
2477 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2478   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2479   { \tl_tail:N \l_tmpa_tl }
2480   \tl_if_single:NnTF \l_tmpa_tl {
2481     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2482       \exp_after:wN \str_set:Nn \exp_after:wN
2483       \l_stex_get_symbol_uri_str \l_tmpa_tl
2484     }{

```

```

2485     % TODO
2486     % tail is not a single group
2487   }
2488 }{
2489   % TODO
2490   % tail is not a single group
2491 }
2492 }

```

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

29.2 Notations

```

2493 <@@=stex_notation>

notation arguments:
2494 \keys_define:nn { stex / notation } {
2495   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2496   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2497   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2498   op       .tl_set:N = \l__stex_notation_op_tl ,
2499   primary .bool_set:N = \l__stex_notation_primary_bool ,
2500   primary .default:n = {true} ,
2501   unknown .code:n = \str_set:Nx
2502             \l__stex_notation_variant_str \l_keys_key_str
2503 }
2504
2505 \cs_new_protected:Nn \stex_notation_args:n {
2506   % \str_clear:N \l__stex_notation_lang_str
2507   \str_clear:N \l__stex_notation_variant_str
2508   \str_clear:N \l__stex_notation_prec_str
2509   \tl_clear:N \l__stex_notation_op_tl
2510   \bool_set_false:N \l__stex_notation_primary_bool
2511
2512   \keys_set:nn { stex / notation } { #1 }
2513 }

\notation

2514 \NewDocumentCommand \notation { s m 0{}} {
2515   \stex_notation_args:n { #3 }
2516   \tl_clear:N \l_stex_symdecl_definiens_tl
2517   \stex_get_symbol:n { #2 }
2518   \tl_set:Nn \l_stex_notation_after_do_tl {
2519     \__stex_notation_final:
2520     \IfBooleanTF#1{
2521       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2522     }{}
2523     \stex_smsmode_do:\ignorespacesandpars
2524   }
2525   \stex_notation_do:nnnnn
2526   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2527   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2528   { \l__stex_notation_variant_str }
2529   { \l__stex_notation_prec_str }

```

```

2530 }
2531 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2532 \seq_new:N \l__stex_notation_precedences_seq
2533 \tl_new:N \l__stex_notation_opprec_tl
2534 \int_new:N \l__stex_notation_currarg_int
2535 \tl_new:N \stex_symbol_after_invokation_tl
2536
2537 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2538   \let\l_stex_current_symbol_str\relax
2539   \seq_clear:N \l__stex_notation_precedences_seq
2540   \tl_clear:N \l__stex_notation_opprec_tl
2541   \str_set:Nx \l__stex_notation_args_str { #1 }
2542   \str_set:Nx \l__stex_notation_arity_str { #2 }
2543   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2544   \str_set:Nx \l__stex_notation_prec_str { #4 }
2545
2546   % precedences
2547   \str_if_empty:NTF \l__stex_notation_prec_str {
2548     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2549       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2550     }{
2551       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2552     }
2553   } {
2554     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2555       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2556       \int_step_inline:nn { \l__stex_notation_arity_str } {
2557         \exp_args:NNo
2558         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2559       }
2560     }{
2561       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2562       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2563         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2564         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2565           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2566             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2567           \seq_map_inline:Nn \l_tmpa_seq {
2568             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2569           }
2570         }
2571       }{
2572         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2573           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2574         }{
2575           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2576         }
2577       }
2578     }
2579   }

```

```

2580
2581 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2582 \int_step_inline:nn { \l__stex_notation_arity_str } {
2583   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2584     \exp_args:NNo
2585     \seq_put_right:No \l__stex_notation_precedences_seq {
2586       \l__stex_notation_opprec_tl
2587     }
2588   }
2589 }
2590 \tl_clear:N \l_stex_notation_dummyargs_tl
2591
2592 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2593   \exp_args:NNe
2594   \cs_set:Npn \l_stex_notation_macrocode_cs {
2595     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2596     { \l__stex_notation_suffix_str }
2597     { \l__stex_notation_opprec_tl }
2598     { \exp_not:n { #5 } }
2599   }
2600   \l_stex_notation_after_do_tl
2601 }{
2602   \str_if_in:NnTF \l__stex_notation_args_str b {
2603     \exp_args:Nne \use:nn
2604     {
2605       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2606       \cs_set:Npn \l__stex_notation_arity_str } { {
2607         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2608         { \l__stex_notation_suffix_str }
2609         { \l__stex_notation_opprec_tl }
2610         { \exp_not:n { #5 } }
2611       }}
2612   }{
2613     \str_if_in:NnTF \l__stex_notation_args_str B {
2614       \exp_args:Nne \use:nn
2615       {
2616         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2617         \cs_set:Npn \l__stex_notation_arity_str } { {
2618           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2619           { \l__stex_notation_suffix_str }
2620           { \l__stex_notation_opprec_tl }
2621           { \exp_not:n { #5 } }
2622         } }
2623     }{
2624       \exp_args:Nne \use:nn
2625       {
2626         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2627         \cs_set:Npn \l__stex_notation_arity_str } { {
2628           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2629           { \l__stex_notation_suffix_str }
2630           { \l__stex_notation_opprec_tl }
2631           { \exp_not:n { #5 } }
2632         } }
2633     }

```

```

2634     }
2635
2636     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2637     \int_zero:N \l__stex_notation_currarg_int
2638     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2639     \__stex_notation_arguments:
2640   }
2641 }

```

(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

```

2642 \cs_new_protected:Nn \__stex_notation_arguments: {
2643   \int_incr:N \l__stex_notation_currarg_int
2644   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2645     \l_stex_notation_after_do_tl
2646   }{
2647     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2648     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2649     \str_if_eq:VnTF \l_tmpa_str a {
2650       \__stex_notation_argument_assoc:nn{a}
2651     }{
2652       \str_if_eq:VnTF \l_tmpa_str B {
2653         \__stex_notation_argument_assoc:nn{B}
2654       }{
2655         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2656         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2657           { \stex_term_math_arg:nnn
2658             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2659             { \l_tmpb_str }
2660             { ###\int_use:N \l__stex_notation_currarg_int }
2661           }
2662         }
2663         \__stex_notation_arguments:
2664       }
2665     }
2666   }
2667 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2668 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2669
2670   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2671     {\l__stex_notation_arity_str}{
2672       #2
2673     }
2674   \int_zero:N \l_tmpa_int
2675   \tl_clear:N \l_tmpa_tl
2676   \str_map_inline:Nn \l__stex_notation_args_str {
2677     \int_incr:N \l_tmpa_int
2678     \tl_put_right:Nx \l_tmpa_tl {
2679       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```



```

2680 \str_if_eq:nnTF {##1}{B}{ } }{
2681   {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2682   }
2683 }
2684 }
2685 }
2686 \exp_after:wN\exp_after:wN\exp_after:wN \def
2687 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2688 \exp_after:wN\exp_after:wN\exp_after:wN ##
2689 \exp_after:wN\exp_after:wN\exp_after:wN 1
2690 \exp_after:wN\exp_after:wN\exp_after:wN ##
2691 \exp_after:wN\exp_after:wN\exp_after:wN 2
2692 \exp_after:wN\exp_after:wN\exp_after:wN {
2693   \exp_after:wN \exp_after:wN \exp_after:wN
2694   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2695     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2696   }
2697 }
2698
2699 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2700 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2701   \_stex_term_math_assoc_arg:nnnn
2702   { #1\int_use:N \l__stex_notation_currarg_int }
2703   { \l_tmpa_str }
2704   { #####\int_use:N \l__stex_notation_currarg_int }
2705   { \l_tmpa_cs {####1} {####2} }
2706 } }
2707 \__stex_notation_arguments:
2708 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2709 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2710   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2711   \cs_set_nopar:Npn {#3}{#4}
2712   \tl_if_empty:nF {#5}{
2713     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2714   }
2715   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2716     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2717   }
2718 }
2719
2720 \cs_new_protected:Nn \_stex_notation_final: {
2721
2722   \stex_execute_in_module:x {
2723     \_stex_notation_restore_notation:nnnnn
2724     {\l_stex_get_symbol_uri_str}
2725     {\l_stex_notation_suffix_str}
2726     {\l__stex_notation_arity_str}
2727     {
2728       \exp_after:wN \exp_after:wN \exp_after:wN
2729       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2730     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2731   }
2732   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2733 }
2734
2735 \stex_debug:nn{symbols}{
2736   Notation~\l__stex_notation_suffix_str
2737   ~for~\l_stex_get_symbol_uri_str^^J
2738   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2739   Argument~precedences:~
2740     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2741   Notation: \cs_meaning:c {
2742     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2743     \l__stex_notation_suffix_str
2744     _cs
2745   }
2746 }
2747 % HTML annotations
2748 \stex_if_do_html:T {
2749   \stex_annotate_invisible:nnn { notation }
2750   { \l_stex_get_symbol_uri_str } {
2751     \stex_annotate_invisible:nnn { notationfragment }
2752     { \l__stex_notation_suffix_str }{}
2753     \stex_annotate_invisible:nnn { precedence }
2754     { \l__stex_notation_prec_str }{}
2755
2756     \int_zero:N \l_tmpa_int
2757     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2758     \tl_clear:N \l_tmpa_tl
2759     \int_step_inline:nn { \l__stex_notation_arity_str }{
2760       \int_incr:N \l_tmpa_int
2761       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2762       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2763       \str_if_eq:VnTF \l_tmpb_str a {
2764         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2765           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2766           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2767         } }
2768       }{
2769         \str_if_eq:VnTF \l_tmpb_str B {
2770           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2771             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2772             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2773           } }
2774         }{
2775           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2776             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2777           } }
2778       }
2779     }
2780   }
2781   \stex_annotate_invisible:nnn { notationcomp }{}{
2782     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2783     $ \exp_args:Nno \use:nn { \use:c {

```

```

2784         stex_notation_ \l_stex_current_symbol_str
2785         \c_hash_str \l__stex_notation_suffix_str _cs
2786     } } { \l_tmpa_tl } $
2787 }
2788 }
2789 }
2790 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2791 \keys_define:nn { stex / setnotation } {
2792   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2793   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2794   unknown .code:n      = \str_set:Nx
2795     \l__stex_notation_variant_str \l_keys_key_str
2796 }
2797
2798 \cs_new_protected:Nn \stex_setnotation_args:n {
2799   % \str_clear:N \l__stex_notation_lang_str
2800   \str_clear:N \l__stex_notation_variant_str
2801   \keys_set:nn { stex / setnotation } { #1 }
2802 }
2803
2804 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2805   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2806     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2807     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2808   }
2809 }
2810
2811 \cs_new_protected:Nn \stex_setnotation:n {
2812   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2813     { \l__stex_notation_variant_str }{
2814     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2815     \stex_debug:nn {notations}{
2816       Setting~default~notation~
2817       {\l__stex_notation_variant_str }~for~
2818       #1 \\
2819       \expandafter\meaning\csname
2820       l_stex_symdecl_#1_notations\endcsname
2821     }
2822   }{
2823     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2824   }
2825 }
2826
2827 \NewDocumentCommand \setnotation {m m} {
2828   \stex_get_symbol:n { #1 }
2829   \stex_setnotation_args:n { #2 }
2830   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2831   \stex_smsmode_do:\ignorespacesandpars
2832 }
2833

```

```

2834 \cs_new_protected:Nn \stex_copy_notations:nn {
2835   \stex_debug:nn {notations}{
2836     Copying~notations~from~#2~to~#1\\
2837     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2838   }
2839   \tl_clear:N \l_tmpa_tl
2840   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2841     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2842   }
2843   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2844     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2845     \edef \l_tmpa_tl {
2846       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2847       \exp_after:wN\exp_after:wN\exp_after:wN {
2848         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2849       }
2850     }
2851
2852     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2853     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2854     \exp_after:wN { \l_tmpa_tl }
2855
2856     \edef \l_tmpa_tl {
2857       \exp_after:wN \exp_not:n \exp_after:wN {
2858         \l_tmpa_tl {##### 1}{##### 2}
2859       }
2860     }
2861
2862     \stex_execute_in_module:x {
2863       \__stex_notation_restore_notation:nnnnn
2864       {#1}{##1}
2865       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2866       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2867       {
2868         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2869           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2870         }
2871       }
2872     }
2873   }
2874 }
2875
2876 \NewDocumentCommand \copynotation {m m} {
2877   \stex_get_symbol:n { #1 }
2878   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2879   \stex_get_symbol:n { #2 }
2880   \exp_args:Noo
2881   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2882   \stex_smsmode_do:\ignorespacesandpars
2883 }
2884

```

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

\symdef

```
2885 \keys_define:nn { stex / symdef } {
2886   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2887   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2888   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2889   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2890   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2891   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2892   op        .tl_set:N    = \l__stex_notation_op_tl ,
2893   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2894   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2895   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2896   assoc     .choices:nn =
2897     {bin,binl,binr,pre,conj,pwconj}
2898     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2899   unknown   .code:n      = \str_set:Nx
2900     \l__stex_notation_variant_str \l_keys_key_str
2901 }
2902
2903 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2904   \str_clear:N \l_stex_symdecl_name_str
2905   \str_clear:N \l_stex_symdecl_args_str
2906   \str_clear:N \l_stex_symdecl_assoctype_str
2907   \str_clear:N \l_stex_symdecl_reorder_str
2908   \bool_set_false:N \l_stex_symdecl_local_bool
2909   \tl_clear:N \l_stex_symdecl_type_tl
2910   \tl_clear:N \l_stex_symdecl_definiens_tl
2911   % \str_clear:N \l__stex_notation_lang_str
2912   \str_clear:N \l__stex_notation_variant_str
2913   \str_clear:N \l__stex_notation_prec_str
2914   \tl_clear:N \l__stex_notation_op_tl
2915
2916   \keys_set:nn { stex / symdef } { #1 }
2917 }
2918
2919 \NewDocumentCommand \symdef { m O{} } {
2920   \__stex_notation_symdef_args:n { #2 }
2921   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2922   \stex_symdecl_do:n { #1 }
2923   \tl_set:Nn \l_stex_notation_after_do_tl {
2924     \__stex_notation_final:
2925     \stex_smsmode_do:\ignorespacesandpars
2926   }
2927   \str_set:Nx \l_stex_get_symbol_uri_str {
2928     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2929   }
2930   \exp_args:Nx \stex_notation_do:nnnnn
2931     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2932     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2933     { \l__stex_notation_variant_str }
2934     { \l__stex_notation_prec_str }
2935 }
2936 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3 Variables

```

2937 <@@=stex_variables>
2938
2939 \keys_define:nn { stex / vardef } {
2940   name      .str_set_x:N = \l__stex_variables_name_str ,
2941   args      .str_set_x:N = \l__stex_variables_args_str ,
2942   type      .tl_set:N    = \l__stex_variables_type_tl ,
2943   def       .tl_set:N    = \l__stex_variables_def_tl ,
2944   op        .tl_set:N    = \l__stex_variables_op_tl ,
2945   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2946   assoc     .choices:nn =
2947     {bin,binl,binr,pre,conj,pwconj}
2948     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2949   bind      .choices:nn =
2950     {forall,exists}
2951     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2952 }
2953
2954 \cs_new_protected:Nn \__stex_variables_args:n {
2955   \str_clear:N \l__stex_variables_name_str
2956   \str_clear:N \l__stex_variables_args_str
2957   \str_clear:N \l__stex_variables_prec_str
2958   \str_clear:N \l__stex_variables_assoctype_str
2959   \str_clear:N \l__stex_variables_bind_str
2960   \tl_clear:N \l__stex_variables_type_tl
2961   \tl_clear:N \l__stex_variables_def_tl
2962   \tl_clear:N \l__stex_variables_op_tl
2963
2964   \keys_set:nn { stex / vardef } { #1 }
2965 }
2966
2967 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2968   \__stex_variables_args:n {#2}
2969   \str_if_empty:NT \l__stex_variables_name_str {
2970     \str_set:Nx \l__stex_variables_name_str { #1 }
2971   }
2972   \prop_clear:N \l_tmpa_prop
2973   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2974
2975   \int_zero:N \l_tmpb_int
2976   \bool_set_true:N \l_tmpa_bool
2977   \str_map_inline:Nn \l__stex_variables_args_str {
2978     \token_case_meaning:NnF ##1 {
2979       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2980       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2981       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2982       {\tl_to_str:n a} {
2983         \bool_set_false:N \l_tmpa_bool
2984         \int_incr:N \l_tmpb_int
2985       }

```

```

2986     {\tl_to_str:n B} {
2987       \bool_set_false:N \l_tmpa_bool
2988       \int_incr:N \l_tmpb_int
2989     }
2990   }{
2991     \msg_error:nxxx{stex}{error/wrongargs}{
2992       variable~\l__stex_variables_name_str
2993     }{##1}
2994   }
2995 }
2996 \bool_if:NTF \l_tmpa_bool {
2997   % possibly numeric
2998   \str_if_empty:NTF \l__stex_variables_args_str {
2999     \prop_put:Nnn \l_tmpa_prop { args } {}
3000     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3001   }{
3002     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3003     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3004     \str_clear:N \l_tmpa_str
3005     \int_step_inline:nn \l_tmpa_int {
3006       \str_put_right:Nn \l_tmpa_str i
3007     }
3008     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3009     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3010   }
3011 } {
3012   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3013   \prop_put:Nnx \l_tmpa_prop { arity }
3014     { \str_count:N \l__stex_variables_args_str }
3015 }
3016 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
3017 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3018
3019 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3020
3021 \tl_if_empty:NF \l__stex_variables_op_tl {
3022   \cs_set:cpx {
3023     stex_var_op_notation_\l__stex_variables_name_str_cs
3024   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3025 }
3026
3027 \tl_set:Nn \l_stex_notation_after_do_tl {
3028   \exp_args:Nne \use:nn {
3029     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
3030     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3031   } {{
3032     \exp_after:wN \exp_after:wN \exp_after:wN
3033     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3034     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
3035   }}
3036 \stex_if_do_html:T {
3037   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3038     \stex_annotate_invisible:nnn { precedence }
3039     { \l__stex_variables_prec_str }{}

```

```

3040 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{-}{\l__
3041 \stex_annotate_invisible:nnn{args}{-}{ \l__stex_variables_args_str }
3042 \stex_annotate_invisible:nnn{macroname}{#1}{-}
3043 \tl_if_empty:NF \l__stex_variables_def_tl {
3044 \stex_annotate_invisible:nnn{definiens}{-}
3045 {\l__stex_variables_def_tl$}
3046 }
3047 \str_if_empty:NF \l__stex_variables_assoctype_str {
3048 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{-}
3049 }
3050 \str_if_empty:NF \l__stex_variables_bind_str {
3051 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{-}
3052 }
3053 \int_zero:N \l_tmpa_int
3054 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3055 \tl_clear:N \l_tmpa_tl
3056 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3057 \int_incr:N \l_tmpa_int
3058 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3059 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3060 \str_if_eq:VnTF \l_tmpb_str a {
3061 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3062 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{-} ,
3063 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{-}
3064 } }
3065 }{
3066 \str_if_eq:VnTF \l_tmpb_str B {
3067 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3068 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{-} ,
3069 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{-}
3070 } }
3071 }{
3072 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3073 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{-}
3074 } }
3075 }
3076 }
3077 }
3078 \stex_annotate_invisible:nnn { notationcomp }{-}{
3079 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3080 $ \exp_args:Nno \use:nn { \use:c {
3081 stex_var_notation_\l__stex_variables_name_str_cs
3082 } } { \l_tmpa_tl } $
3083 }
3084 }
3085 }\ignorespacesandpars
3086 }
3087
3088 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3089 }
3090
3091 \cs_new:Nn \_stex_reset:N {
3092 \tl_if_exist:NTF #1 {
3093 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```



```

3094 }{
3095   \let \exp_not:N #1 \exp_not:N \undefined
3096 }
3097 }
3098
3099 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3100   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3101   \exp_args:Nnx \use:nn {
3102     % TODO
3103     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3104       #2
3105     }
3106   }{
3107     \stex_reset:N \varnot
3108     \stex_reset:N \vartype
3109     \stex_reset:N \vardefi
3110   }
3111 }
3112
3113 \NewDocumentCommand \vardef { s } {
3114   \IfBooleanTF#1 {
3115     \__stex_variables_do_complex:nn
3116   }{
3117     \__stex_variables_do_simple:nnn
3118   }
3119 }
3120
3121 \NewDocumentCommand \svar { 0{} m }{
3122   \tl_if_empty:nTF {#1}{
3123     \str_set:Nn \l_tmpa_str { #2 }
3124   }{
3125     \str_set:Nn \l_tmpa_str { #1 }
3126   }
3127   \stex_term_omv:nn {
3128     var://\l_tmpa_str
3129   }{
3130     \exp_args:Nnx \use:nn {
3131       \def\comp{\_varcomp}
3132       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3133       \comp{ #2 }
3134     }{
3135       \stex_reset:N \comp
3136       \stex_reset:N \l_stex_current_symbol_str
3137     }
3138   }
3139 }
3140
3141
3142
3143 \keys_define:nn { stex / varseq } {
3144   name .str_set_x:N = \l__stex_variables_name_str ,
3145   args .int_set:N = \l__stex_variables_args_int ,
3146   type .tl_set:N = \l__stex_variables_type_tl ,
3147   mid .tl_set:N = \l__stex_variables_mid_tl ,

```

```

3148   bind      .choices:nn      =
3149           {forall,exists}
3150           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3151   }
3152
3153   \cs_new_protected:Nn \__stex_variables_seq_args:n {
3154     \str_clear:N \l__stex_variables_name_str
3155     \int_set:Nn \l__stex_variables_args_int 1
3156     \tl_clear:N \l__stex_variables_type_tl
3157     \str_clear:N \l__stex_variables_bind_str
3158
3159     \keys_set:nn { stex / varseq } { #1 }
3160   }
3161
3162   \NewDocumentCommand \varseq {m O{}} m m m m {
3163     \__stex_variables_seq_args:n { #2 }
3164     \str_if_empty:NT \l__stex_variables_name_str {
3165       \str_set:Nx \l__stex_variables_name_str { #1 }
3166     }
3167     \prop_clear:N \l_tmpa_prop
3168     \prop_put:Nnx \l_tmpa_prop { arity } {\int_use:N \l__stex_variables_args_int}
3169
3170     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3171     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3172       \msg_error:nnxx{stex}{error/seqlength}
3173       {\int_use:N \l__stex_variables_args_int}
3174       {\seq_count:N \l_tmpa_seq}
3175     }
3176     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3177     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3178       \msg_error:nnxx{stex}{error/seqlength}
3179       {\int_use:N \l__stex_variables_args_int}
3180       {\seq_count:N \l_tmpb_seq}
3181     }
3182     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3183     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3184
3185     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3186     \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
3187
3188     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3189     \int_step_inline:nn \l__stex_variables_args_int {
3190       \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3191     }
3192     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3193     \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3194     \tl_if_empty:NF \l__stex_variables_mid_tl {
3195       \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3196       \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3197     }
3198     \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3199     \int_step_inline:nn \l__stex_variables_args_int {
3200       \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3201     }

```

```

3202 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3203 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3204
3205
3206 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3207
3208 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3209
3210 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3211
3212 \int_step_inline:nn \l__stex_variables_args_int {
3213   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3214     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3215   }}
3216 }
3217
3218 \tl_set:Nx \l_tmpa_tl {
3219   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3220     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3221   }
3222 }
3223
3224 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3225
3226 \exp_args:Nno \use:nn {
3227   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3228   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3229
3230 \stex_debug:nn{sequences}{New~Sequence:~
3231   \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3232   \prop_to_keyval:N \l_tmpa_prop
3233 }
3234 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3235   \tl_if_empty:NF \l__stex_variables_type_tl {
3236     \stex_annotate:nnn {type}{\}{\seqtype\l__stex_variables_type_tl$}
3237   }
3238   \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3239   \str_if_empty:NF \l__stex_variables_bind_str {
3240     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3241   }
3242 }}
3243
3244 \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3245 \ignorespacesandpars
3246 }
3247
3248 </package>

```

Chapter 30

STEX -Terms Implementation

```
3249 <*package>
3250
3251 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3252
3253 <@@=stex_terms>
3254
3255 Warnings and error messages
3256 \msg_new:nnn{stex}{error/nonotation}{
3257   Symbol~#1~invoked,~but~has~no~notation#2!
3258 }
3259 \msg_new:nnn{stex}{error/notationarg}{
3260   Error~in~parsing~notation~#1
3261 }
3262 \msg_new:nnn{stex}{error/noop}{
3263   Symbol~#1~has~no~operator~notation~for~notation~#2
3264 }
3265 \msg_new:nnn{stex}{error/notallowed}{
3266   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3267 }
3268 \msg_new:nnn{stex}{error/doubleargument}{
3269   Argument~#1~of~symbol~#2~already~assigned
3270 }
3271 \msg_new:nnn{stex}{error/overarity}{
3272   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3273 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3273
3274
3275 \bool_new:N \l_stex_allow_semantic_bool
3276 \bool_set_true:N \l_stex_allow_semantic_bool
3277
```

```

3278 \cs_new_protected:Nn \stex_invoke_symbol:n {
3279   \bool_if:NTF \l_stex_allow_semantic_bool {
3280     \str_if_eq:eeF {
3281       \prop_item:cn {
3282         l_stex_symdecl_#1_prop
3283       }{ deprecate }
3284     }{}{
3285       \msg_warning:nxxx{stex}{warning/deprecated}{
3286         Symbol~#1
3287       }{
3288         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3289       }
3290     }
3291     \if_mode_math:
3292       \exp_after:wN \__stex_terms_invoke_math:n
3293     \else:
3294       \exp_after:wN \__stex_terms_invoke_text:n
3295     \fi: { #1 }
3296   }{
3297     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3298   }
3299 }
3300
3301 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3302   \peek_charcode_remove:NTF ! {
3303     \__stex_terms_invoke_op_custom:nn {#1}
3304   }{
3305     \__stex_terms_invoke_custom:nn {#1}
3306   }
3307 }
3308
3309 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3310   \peek_charcode_remove:NTF ! {
3311     % operator
3312     \peek_charcode_remove:NTF * {
3313       % custom op
3314       \__stex_terms_invoke_op_custom:nn {#1}
3315     }{
3316       % op notation
3317       \peek_charcode:NTF [ {
3318         \__stex_terms_invoke_op_notation:nw {#1}
3319       }{
3320         \__stex_terms_invoke_op_notation:nw {#1}[]
3321       }
3322     }
3323   }{
3324     \peek_charcode_remove:NTF * {
3325       \__stex_terms_invoke_custom:nn {#1}
3326       % custom
3327     }{
3328       % normal
3329       \peek_charcode:NTF [ {
3330         \__stex_terms_invoke_notation:nw {#1}
3331       }{

```

```

3332     \__stex_terms_invoke_notation:nw {#1}[]
3333   }
3334 }
3335 }
3336 }
3337
3338
3339 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3340   \exp_args:Nnx \use:nn {
3341     \def\comp{\_comp}
3342     \str_set:Nn \l_stex_current_symbol_str { #1 }
3343     \bool_set_false:N \l_stex_allow_semantic_bool
3344     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3345       \comp{ #2 }
3346     }
3347   }{
3348     \stex_reset:N \comp
3349     \stex_reset:N \l_stex_current_symbol_str
3350     \bool_set_true:N \l_stex_allow_semantic_bool
3351   }
3352 }
3353
3354 \keys_define:nn { stex / terms } {
3355   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3356   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3357   unknown .code:n = \str_set:Nx
3358     \l_stex_notation_variant_str \l_keys_key_str
3359 }
3360
3361 \cs_new_protected:Nn \__stex_terms_args:n {
3362   % \str_clear:N \l_stex_notation_lang_str
3363   \str_clear:N \l_stex_notation_variant_str
3364
3365   \keys_set:nn { stex / terms } { #1 }
3366 }
3367
3368 \cs_new_protected:Nn \stex_find_notation:nn {
3369   \__stex_terms_args:n { #2 }
3370   \seq_if_empty:cTF {
3371     l_stex_symdecl_ #1 _notations
3372   } {
3373     \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3374   } {
3375     \str_if_empty:NTF \l_stex_notation_variant_str {
3376       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3377     }{
3378       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3379         \l_stex_notation_variant_str
3380       }{
3381         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3382       }{
3383         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3384           ~\l_stex_notation_variant_str
3385         }
3386       }
3387     }
3388   }

```

```

3386     }
3387   }
3388 }
3389 }
3390
3391 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3392   \exp_args:Nnx \use:nn {
3393     \def\comp{\_comp}
3394     \str_set:Nn \l_stex_current_symbol_str { #1 }
3395     \stex_find_notation:nn { #1 }{ #2 }
3396     \bool_set_false:N \l_stex_allow_semantic_bool
3397     \cs_if_exist:cTF {
3398       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3399     }{
3400       \_stex_term_oms:nnn { #1 }{
3401         #1 \c_hash_str \l_stex_notation_variant_str
3402       }{
3403         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3404       }
3405     }{
3406       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3407         \cs_if_exist:cTF {
3408           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3409         }{
3410           \tl_set:Nx \stex_symbol_after_invokation_tl {
3411             \_stex_reset:N \comp
3412             \_stex_reset:N \stex_symbol_after_invokation_tl
3413             \_stex_reset:N \l_stex_current_symbol_str
3414             \bool_set_true:N \l_stex_allow_semantic_bool
3415           }
3416           \def\comp{\_comp}
3417           \str_set:Nn \l_stex_current_symbol_str { #1 }
3418           \bool_set_false:N \l_stex_allow_semantic_bool
3419           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3420         }{
3421           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3422             ~\l_stex_notation_variant_str
3423           }
3424         }
3425       }{
3426         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3427       }
3428     }
3429   }{
3430     \_stex_reset:N \comp
3431     \_stex_reset:N \l_stex_current_symbol_str
3432     \bool_set_true:N \l_stex_allow_semantic_bool
3433   }
3434 }
3435
3436 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3437   \stex_find_notation:nn { #1 }{ #2 }
3438   \cs_if_exist:cTF {
3439     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3440 }{
3441   \tl_set:Nx \stex_symbol_after_invokation_tl {
3442     \_stex_reset:N \comp
3443     \_stex_reset:N \stex_symbol_after_invokation_tl
3444     \_stex_reset:N \l_stex_current_symbol_str
3445     \bool_set_true:N \l_stex_allow_semantic_bool
3446   }
3447   \def\comp{\_comp}
3448   \str_set:Nn \l_stex_current_symbol_str { #1 }
3449   \bool_set_false:N \l_stex_allow_semantic_bool
3450   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3451 }{
3452   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3453     ~\l_stex_notation_variant_str
3454   }
3455 }
3456 }
3457
3458 \prop_new:N \l__stex_terms_custom_args_prop
3459
3460 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3461   \exp_args:Nnx \use:nn {
3462     \bool_set_false:N \l_stex_allow_semantic_bool
3463     \def\comp{\_comp}
3464     \str_set:Nn \l_stex_current_symbol_str { #1 }
3465     \prop_clear:N \l__stex_terms_custom_args_prop
3466     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3467     \prop_get:cnN {
3468       l_stex_symdecl_#1 _prop
3469     }{ args } \l_tmpa_str
3470     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3471     \tl_set:Nn \arg { \__stex_terms_arg: }
3472     \str_if_empty:NTF \l_tmpa_str {
3473       \_stex_term oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3474     }{
3475       \str_if_in:NnTF \l_tmpa_str b {
3476         \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3477       }{
3478         \str_if_in:NnTF \l_tmpa_str B {
3479           \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3480         }{
3481           \_stex_term oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3482         }
3483       }
3484     }
3485     % TODO check that all arguments exist
3486   }{
3487     \_stex_reset:N \l_stex_current_symbol_str
3488     \_stex_reset:N \arg
3489     \_stex_reset:N \comp
3490     \_stex_reset:N \l__stex_terms_custom_args_prop
3491     \bool_set_true:N \l_stex_allow_semantic_bool
3492   }
3493 }

```



```

3494
3495 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3496   \tl_if_empty:nTF {#2}{
3497     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3498     \bool_set_true:N \l_tmpa_bool
3499     \bool_do_while:Nn \l_tmpa_bool {
3500       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3501         \int_incr:N \l_tmpa_int
3502       }{
3503         \bool_set_false:N \l_tmpa_bool
3504       }
3505     }
3506   }{
3507     \int_set:Nn \l_tmpa_int { #2 }
3508   }
3509   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3510   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3511     \msg_error:nnxxx{stex}{error/overarity}
3512     {\int_use:N \l_tmpa_int}
3513     {\l_stex_current_symbol_str}
3514     {\str_count:N \l_tmpa_str}
3515   }
3516   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3517   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3518     \bool_lazy_any:nF {
3519       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3520       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3521     }{
3522       \msg_error:nnxx{stex}{error/doubleargument}
3523       {\int_use:N \l_tmpa_int}
3524       {\l_stex_current_symbol_str}
3525     }
3526   }
3527   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3528   \bool_set_true:N \l_stex_allow_semantic_bool
3529   \IfBooleanTF#1{
3530     \stex_annotate_invisible:n { %TODO
3531       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3532     }
3533   }{ %TODO
3534     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3535   }
3536   \bool_set_false:N \l_stex_allow_semantic_bool
3537 }
3538
3539
3540 \cs_new_protected:Nn \_stex_term_arg:nn {
3541   \bool_set_true:N \l_stex_allow_semantic_bool
3542   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3543   \bool_set_false:N \l_stex_allow_semantic_bool
3544 }
3545
3546 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3547   \exp_args:Nnx \use:nn

```

```

3548 { \int_set:Nn \l__stex_terms_downprec { #2 }
3549   \stex_term_arg:nn { #1 }{ #3 }
3550 }
3551 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3552 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

3553 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3554   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3555   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3556   \tl_if_empty:NTF { #3 }{
3557     \stex_term_math_arg:nnn{#1}{#2}{#3}
3558   }{
3559     \exp_args:Nx \tl_if_empty:NTF { \tl_tail:n{ #3 } }{
3560       \expandafter\if\expandafter\relax\noexpand#3
3561       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3562     }else
3563       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3564     \fi
3565     \l_tmpa_tl
3566   }{
3567     \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3568   }
3569 }
3570 }
3571
3572 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3573   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3574   \str_if_empty:NTF \l_tmpa_str {
3575     \exp_args:Nx \cs_if_eq:NNTF {
3576       \tl_head:N #1
3577     } \stex_invoke_sequence:n {
3578       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3579       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3580       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3581       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3582       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3583         \exp_not:n{\exp_args:Nnx \use:nn} {
3584           \exp_not:n {
3585             \def\comp{\_varcomp}
3586             \str_set:Nn \l_stex_current_symbol_str
3587             } {varseq://\l_tmpa_str}
3588           \exp_not:n{ ##1 }
3589         }{
3590           \exp_not:n {
3591             \stex_reset:N \comp
3592             \stex_reset:N \l_stex_current_symbol_str
3593           }
3594         }
3595       }}}
3596   \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3597   \seq_reverse:N \l_tmpa_seq

```

```

3598     \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3599     \seq_map_inline:Nn \l_tmpa_seq {
3600         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3601             \exp_args:Nno
3602             \l_tmpa_cs { ##1 } \l_tmpa_tl
3603         }
3604     }
3605     \tl_set:Nx \l_tmpa_tl {
3606         \stex_term_omv:nn {varseq://\l_tmpa_str}{
3607             \exp_args:No \exp_not:n \l_tmpa_tl
3608         }
3609     }
3610     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3611 }{
3612     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3613 }
3614 } {
3615     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3616 }
3617 }
3618 }
3619
3620 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3621     \clist_set:Nn \l_tmpa_clist{ #2 }
3622     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3623         \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3624     }{
3625         \clist_reverse:N \l_tmpa_clist
3626         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3627         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3628             \exp_args:No \exp_not:n \l_tmpa_tl
3629         }}
3630         \clist_map_inline:Nn \l_tmpa_clist {
3631             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3632                 \exp_args:Nno
3633                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3634             }
3635         }
3636     }
3637     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3638 }

```

(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
3639 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3640 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3641 \int_new:N \l__stex_terms_downprec
3642 \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`

```
3643 \tl_set:Nn \l__stex_terms_left_bracket_str (
3644 \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

```
3645 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3646   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3647     \bool_set_false:N \l__stex_terms_brackets_done_bool
3648     #2
3649   } {
3650     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3651       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3652         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3653         \dobrackets { #2 }
3654       }
3655     }{ #2 }
3656   }
3657 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
3658 \bool_new:N \l__stex_terms_brackets_done_bool
3659 %\RequirePackage{scalerel}
3660 \cs_new_protected:Npn \dobrackets #1 {
3661   %\ThisStyle{\if D\m@switch
3662   %   \exp_args:Nnx \use:nn
3663   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3664   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3665   %   \else
3666   \exp_args:Nnx \use:nn
3667   {
3668     \bool_set_true:N \l__stex_terms_brackets_done_bool
3669     \int_set:Nn \l__stex_terms_downprec \infprec
3670     \l__stex_terms_left_bracket_str
3671     #1
3672   }
3673   {
3674     \bool_set_false:N \l__stex_terms_brackets_done_bool
3675     \l__stex_terms_right_bracket_str
3676     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3677   }
3678   %\fi}
3679 }
```

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

\withbrackets

```
3680 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3681   \exp_args:Nnx \use:nn
3682   {
3683     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3684     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3685     #3
3686   }
3687   {
3688     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3689     {\l__stex_terms_left_bracket_str}
3690     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3691     {\l__stex_terms_right_bracket_str}
3692   }
3693 }
```

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

\STEXinvisible

```
3694 \cs_new_protected:Npn \STEXinvisible #1 {
3695   \stex_annotate_invisible:n { #1 }
3696 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3697 \cs_new_protected:Nn \_stex_term_oms:nnn {
3698   \stex_annotate:nnn{ OMID }{ #2 }{
3699     #3
3700   }
3701 }
3702
3703 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3704   \__stex_terms_maybe_brackets:nn { #3 }{
3705     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3706   }
3707 }
```

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

_stex_term_math_omv:nn

```
3708 \cs_new_protected:Nn \_stex_term_omv:nn {
3709   \stex_annotate:nnn{ OMV }{ #1 }{
3710     #2
3711   }
3712 }
```

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

_stex_term_math_oma:nnnn

```
3713 \cs_new_protected:Nn \_stex_term_oma:nnn {
3714   \stex_annotate:nnn{ OMA }{ #2 }{
3715     #3
3716   }
```

```

3717 }
3718
3719 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3720   \__stex_terms_maybe_brackets:nn { #3 }{
3721     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3722   }
3723 }

```

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

`\stex_term_math_omb:nnnn`

```

3724 \cs_new_protected:Nn \stex_term_ombind:nnn {
3725   \stex_annotate:nnn{ OMBIND }{ #2 }{
3726     #3
3727   }
3728 }
3729
3730 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3731   \__stex_terms_maybe_brackets:nn { #3 }{
3732     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3733   }
3734 }

```

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

`\symref`
`\symname`

```

3735 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3736
3737 \keys_define:nn { stex / symname } {
3738   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3739   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3740   root     .tl_set_x:N      = \l__stex_terms_root_tl
3741 }
3742
3743 \cs_new_protected:Nn \stex_symname_args:n {
3744   \tl_clear:N \l__stex_terms_post_tl
3745   \tl_clear:N \l__stex_terms_pre_tl
3746   \tl_clear:N \l__stex_terms_root_str
3747   \keys_set:nn { stex / symname } { #1 }
3748 }
3749
3750 \NewDocumentCommand \symref { m m }{
3751   \let\compemph_uri_prev:\compemph@uri
3752   \let\compemph@uri\symrefemph@uri
3753   \STEXsymbol{#1}!\{ #2 }
3754   \let\compemph@uri\compemph_uri_prev:
3755 }
3756
3757 \NewDocumentCommand \synonym { O{} m m }{
3758   \stex_symname_args:n { #1 }
3759   \let\compemph_uri_prev:\compemph@uri
3760   \let\compemph@uri\symrefemph@uri
3761   % TODO
3762   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3763   \let\compemph@uri\compemph_uri_prev:

```

```

3764 }
3765
3766 \NewDocumentCommand \symname { 0{} m }{
3767   \stex_symname_args:n { #1 }
3768   \stex_get_symbol:n { #2 }
3769   \str_set:Nx \l_tmpa_str {
3770     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3771   }
3772   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3773
3774   \let\compemph_uri_prev:\compemph@uri
3775   \let\compemph@uri\symrefemph@uri
3776   \exp_args:NNx \use:nn
3777   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3778     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3779   } }
3780   \let\compemph@uri\compemph_uri_prev:
3781 }
3782
3783 \NewDocumentCommand \Symname { 0{} m }{
3784   \stex_symname_args:n { #1 }
3785   \stex_get_symbol:n { #2 }
3786   \str_set:Nx \l_tmpa_str {
3787     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3788   }
3789   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3790   \let\compemph_uri_prev:\compemph@uri
3791   \let\compemph@uri\symrefemph@uri
3792   \exp_args:NNx \use:nn
3793   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3794     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3795     \l__stex_terms_post_tl
3796   } }
3797   \let\compemph@uri\compemph_uri_prev:
3798 }

```

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

30.3 Notation Components

```

3799 <@@=stex_notationcomps>

\comp
\compemph@uri 3800 \cs_new_protected:Npn \_comp #1 {
\compemph 3801   \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3802     \stex_html_backend:TF {
\defemph@uri 3803       \stex_annotate:nnn { comp }{\l_stex_current_symbol_str }{ #1 }
\symrefemph 3804     }{
\symrefemph@uri 3805       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
\varemp 3806     }
\varemp@uri 3807   }
3808 }
3809
3810 \cs_new_protected:Npn \_varcomp #1 {

```

```

3811 \str_if_empty:NF \l_stex_current_symbol_str {
3812   \stex_html_backend:TF {
3813     \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3814   }{
3815     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3816   }
3817 }
3818 }
3819
3820 \def\comp{\_comp}
3821
3822 \cs_new_protected:Npn \compemph@uri #1 #2 {
3823   \compemph{ #1 }
3824 }
3825
3826
3827 \cs_new_protected:Npn \compemph #1 {
3828   #1
3829 }
3830
3831 \cs_new_protected:Npn \defemph@uri #1 #2 {
3832   \defemph{#1}
3833 }
3834
3835 \cs_new_protected:Npn \defemph #1 {
3836   \textbf{#1}
3837 }
3838
3839 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3840   \symrefemph{#1}
3841 }
3842
3843 \cs_new_protected:Npn \symrefemph #1 {
3844   \emph{#1}
3845 }
3846
3847 \cs_new_protected:Npn \varemp@uri #1 #2 {
3848   \varemp{#1}
3849 }
3850
3851 \cs_new_protected:Npn \varemp #1 {
3852   #1
3853 }

```

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

\ellipses

```

3854 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3855 \bool_new:N \l_stex_inparray_bool
\parrayline 3856 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3857 \NewDocumentCommand \parray { m m } {
\parraycell

```



```

3858 \begingroup
3859 \bool_set_true:N \l_stex_inarray_bool
3860 \begin{array}{#1}
3861 #2
3862 \end{array}
3863 \endgroup
3864 }
3865
3866 \NewDocumentCommand \prmatrix { m } {
3867 \begingroup
3868 \bool_set_true:N \l_stex_inarray_bool
3869 \begin{matrix}
3870 #1
3871 \end{matrix}
3872 \endgroup
3873 }
3874
3875 \def \maybepline {
3876 \bool_if:NT \l_stex_inarray_bool {\hline}
3877 }
3878
3879 \def \parrayline #1 #2 {
3880 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3881 }
3882
3883 \def \pmrow #1 { \parrayline{}{ #1 } }
3884
3885 \def \parraylineh #1 #2 {
3886 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3887 }
3888
3889 \def \parraycell #1 {
3890 #1 \bool_if:NT \l_stex_inarray_bool {&}
3891 }

```

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

30.4 Variables

```

3892 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3893 \cs_new_protected:Nn \stex_invoke_variable:n {
3894 \if_mode_math:
3895 \exp_after:wN \__stex_variables_invoke_math:n
3896 \else:
3897 \exp_after:wN \__stex_variables_invoke_text:n
3898 \fi: {#1}
3899 }
3900
3901 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3902 %TODO
3903 }
3904

```

```

3905
3906 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3907   \peek_charcode_remove:NTF ! {
3908     \peek_charcode_remove:NTF ! {
3909       \peek_charcode:NTF [ {
3910         \__stex_variables_invoke_op_custom:nw
3911       }{
3912         % TODO throw error
3913       }
3914     }{
3915       \__stex_variables_invoke_op:n { #1 }
3916     }
3917   }{
3918     \peek_charcode_remove:NTF * {
3919       \__stex_variables_invoke_text:n { #1 }
3920     }{
3921       \__stex_variables_invoke_math_ii:n { #1 }
3922     }
3923   }
3924 }
3925
3926 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3927   \cs_if_exist:cTF {
3928     stex_var_op_notation_ #1 _cs
3929   }{
3930     \exp_args:Nnx \use:nn {
3931       \def\comp{\_varcomp}
3932       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3933       \_stex_term_omv:nn { var://#1 }{
3934         \use:c{stex_var_op_notation_ #1 _cs }
3935       }
3936     }{
3937       \_stex_reset:N \comp
3938       \_stex_reset:N \l_stex_current_symbol_str
3939     }
3940   }{
3941     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3942       \__stex_variables_invoke_math_ii:n {#1}
3943     }{
3944       \msg_error:nxxx{stex}{error/noop}{variable~#1}{}
3945     }
3946   }
3947 }
3948
3949 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3950   \cs_if_exist:cTF {
3951     stex_var_notation_#1_cs
3952   }{
3953     \tl_set:Nx \stex_symbol_after_invokation_tl {
3954       \_stex_reset:N \comp
3955       \_stex_reset:N \stex_symbol_after_invokation_tl
3956       \_stex_reset:N \l_stex_current_symbol_str
3957       \bool_set_true:N \l_stex_allow_semantic_bool
3958     }

```

```

3959     \def\comp{\_varcomp}
3960     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3961     \bool_set_false:N \l_stex_allow_semantic_bool
3962     \use:c{stex_var_notation_#1_cs}
3963   }{
3964     \msg_error:nxx{stex}{error/nonotation}{variable-#1}{s}
3965   }
3966 }

```

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

30.5 Sequences

```

3967 <@@=stex_sequences>
3968
3969 \cs_new_protected:Nn \stex_invoke_sequence:n {
3970   \peek_charcode_remove:NTF ! {
3971     \_stex_term_omv:nn {varseq://#1}{
3972       \exp_args:Nnx \use:nn {
3973         \def\comp{\_varcomp}
3974         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3975         \prop_item:cn{stex_varseq_#1_prop}{notation}
3976       }{
3977         \_stex_reset:N \comp
3978         \_stex_reset:N \l_stex_current_symbol_str
3979       }
3980     }
3981   }{
3982     \bool_set_false:N \l_stex_allow_semantic_bool
3983     \def\comp{\_varcomp}
3984     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3985     \tl_set:Nx \stex_symbol_after_invokation_tl {
3986       \_stex_reset:N \comp
3987       \_stex_reset:N \stex_symbol_after_invokation_tl
3988       \_stex_reset:N \l_stex_current_symbol_str
3989       \bool_set_true:N \l_stex_allow_semantic_bool
3990     }
3991     \use:c { stex_varseq_#1_cs }
3992   }
3993 }
3994 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3995 <*package>
3996
3997 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3998
    Warnings and error messages
3999 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4000     Symbol~#1~can~not~be~assigned~in~copymodule~#2
4001 }
4002 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4003     Symbol~#1~not~assigned~in~interpretmodule~#2
4004 }
4005
4006 \msg_new:nnn{stex}{error/unknownstructure}{
4007     No~structure~#1~found!
4008 }
4009
4010 \msg_new:nnn{stex}{error/unknownfield}{
4011     No~field~#1~in~instance~#2~found!\#3
4012 }
4013
4014 \msg_new:nnn{stex}{error/keyval}{
4015     Invalid~key=value~pair:#1
4016 }
4017 \msg_new:nnn{stex}{error/instantiate/missing}{
4018     Assignments~missing~in~instantiate:~#1
4019 }
4020 \msg_new:nnn{stex}{error/incompatible}{
4021     Incompatible~signature:~#1~(#2)~and~#3~(#4)
4022 }
4023
```

31.1 Imports with modification

```

4024 <@@=stex_copymodule>
4025 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4026   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4027     \tl_set:Nn \l_tmpa_tl { #1 }
4028     \__stex_copymodule_get_symbol_from_cs:
4029   }{
4030     % argument is a string
4031     % is it a command name?
4032     \cs_if_exist:cTF { #1 }{
4033       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4034       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4035       \str_if_empty:NTF \l_tmpa_str {
4036         \exp_args:Nx \cs_if_eq:NNTF {
4037           \tl_head:N \l_tmpa_tl
4038         } \stex_invoke_symbol:n {
4039           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4040         }{
4041           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4042         }
4043       } {
4044         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4045       }
4046     }{
4047       % argument is not a command name
4048       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4049       % \l_stex_all_symbols_seq
4050     }
4051   }
4052 }
4053
4054 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4055   \str_set:Nn \l_tmpa_str { #1 }
4056   \bool_set_false:N \l_tmpa_bool
4057   \bool_if:NF \l_tmpa_bool {
4058     \tl_set:Nn \l_tmpa_tl {
4059       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4060     }
4061     \str_set:Nn \l_tmpa_str { #1 }
4062     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4063     \seq_map_inline:Nn #2 {
4064       \str_set:Nn \l_tmpb_str { ##1 }
4065       \str_if_eq:eeT { \l_tmpa_str } {
4066         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4067       } {
4068         \seq_map_break:n {
4069           \tl_set:Nn \l_tmpa_tl {
4070             \str_set:Nn \l_stex_get_symbol_uri_str {
4071               ##1
4072             }
4073           }
4074         }
4075       }

```

```

4076     }
4077     \l_tmpa_tl
4078   }
4079 }
4080
4081 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4082   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4083     { \tl_tail:N \l_tmpa_tl }
4084   \tl_if_single:NTF \l_tmpa_tl {
4085     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4086       \exp_after:wN \str_set:Nn \exp_after:wN
4087         \l_stex_get_symbol_uri_str \l_tmpa_tl
4088       \__stex_copymodule_get_symbol_check:n { #1 }
4089     }{
4090       % TODO
4091       % tail is not a single group
4092     }
4093   }{
4094     % TODO
4095     % tail is not a single group
4096   }
4097 }
4098
4099 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4100   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4101     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4102       :~\seq_use:Nn #1 {,~}
4103     }
4104   }
4105 }
4106
4107 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4108   % import module
4109   \stex_import_module_uri:nn { #1 } { #2 }
4110   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4111   \stex_import_require_module:nnnn
4112     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4113     { \l_stex_import_path_str } { \l_stex_import_name_str }
4114
4115   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4116   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4117
4118   % fields
4119   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4120   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4121     \seq_map_inline:cn {c_stex_module_##1_constants}{
4122       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4123         ##1 ? ####1
4124       }
4125     }
4126   }
4127
4128   % setup prop
4129   \seq_clear:N \l_tmpa_seq

```

```

4130 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4131   name      = \l_stex_current_copymodule_name_str ,
4132   module    = \l_stex_current_module_str ,
4133   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4134   includes  = \l_tmpa_seq %,
4135 % fields    = \l_tmpa_seq
4136 }
4137 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4138   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4139 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4140 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4141
4142 \stex_if_do_html:T {
4143   \begin{stex_annotate_env} {#4} {
4144     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4145   }
4146   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4147 }
4148 }
4149
4150 \cs_new_protected:Nn \stex_copymodule_end:n {
4151   % apply to every field
4152   \def \l_tmpa_cs ##1 ##2 {#1}
4153
4154   \tl_clear:N \__stex_copymodule_module_tl
4155   \tl_clear:N \__stex_copymodule_exec_tl
4156
4157   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4158   \seq_clear:N \__stex_copymodule_fields_seq
4159
4160   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4161     \seq_map_inline:cn {c_stex_module_##1_constants}{
4162
4163       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4164       \l_tmpa_cs{##1}{####1}
4165
4166       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4167         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4168         \stex_if_do_html:T {
4169           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4170             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4171           }
4172         }
4173       }{
4174         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4175       }
4176
4177       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4178       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4179       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4180
4181       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4182         \stex_if_do_html:T {
4183           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4184         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4185     }
4186 }
4187 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4188 }
4189
4190 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4191 \tl_put_right:Nx \__stex_copymodule_module_tl {
4192     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4193     \prop_set_from_keyval:cn {
4194         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4195     }{
4196         \prop_to_keyval:N \l_tmpa_prop
4197     }
4198 }
4199
4200 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4201     \stex_if_do_html:T {
4202         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4203             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4204         }
4205     }
4206     \tl_put_right:Nx \__stex_copymodule_module_tl {
4207         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4208             \stex_invoke_symbol:n {
4209                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4210             }
4211         }
4212     }
4213 }
4214
4215 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4216
4217 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4218     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4219 }
4220
4221 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4222     \stex_if_do_html:TF{
4223         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4224     }{
4225         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4226     }
4227 }
4228 }
4229 }
4230
4231
4232 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4233 \tl_put_left:Nx \__stex_copymodule_module_tl {
4234     \prop_set_from_keyval:cn {
4235         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4236     }{
4237         \prop_to_keyval:N \l_stex_current_copymodule_prop

```



```

4238     }
4239   }
4240
4241   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4242     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4243   }
4244
4245   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4246   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4247   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4248
4249   \__stex_copymodule_exec_tl
4250   \stex_if_do_html:T {
4251     \end{stex_annotate_env}
4252   }
4253 }
4254
4255 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4256   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4257   \stex_deactivate_macro:Nn \symdecl {module~environments}
4258   \stex_deactivate_macro:Nn \symdef {module~environments}
4259   \stex_deactivate_macro:Nn \notation {module~environments}
4260   \stex_reactivate_macro:N \assign
4261   \stex_reactivate_macro:N \renamedekl
4262   \stex_reactivate_macro:N \donotcopy
4263   \stex_smsmode_do:
4264 }{
4265   \stex_copymodule_end:n {}
4266 }
4267
4268 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4269   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4270   \stex_deactivate_macro:Nn \symdecl {module~environments}
4271   \stex_deactivate_macro:Nn \symdef {module~environments}
4272   \stex_deactivate_macro:Nn \notation {module~environments}
4273   \stex_reactivate_macro:N \assign
4274   \stex_reactivate_macro:N \renamedekl
4275   \stex_reactivate_macro:N \donotcopy
4276   \stex_smsmode_do:
4277 }{
4278   \stex_copymodule_end:n {
4279     \tl_if_exist:cF {
4280       l__stex_copymodule_copymodule_##1?##2_def_tl
4281     }{
4282       \str_if_eq:eeF {
4283         \prop_item:cn{
4284           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4285         }{ true }{
4286           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4287             ##1?##2
4288           }{\l_stex_current_copymodule_name_str}
4289         }
4290       }
4291     }

```

```

4292 }
4293
4294 \iffalse \begin{stex_annotate_env} \fi
4295 \NewDocumentEnvironment {realization} { 0 } { m } {
4296   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4297   \stex_deactivate_macro:Nn \symdecl {module~environments}
4298   \stex_deactivate_macro:Nn \symdef {module~environments}
4299   \stex_deactivate_macro:Nn \notation {module~environments}
4300   \stex_reactivate_macro:N \donotcopy
4301   \stex_reactivate_macro:N \assign
4302   \stex_smsmode_do:
4303 } {
4304   \stex_import_module_uri:nn { #1 } { #2 }
4305   \tl_clear:N \__stex_copymodule_exec_tl
4306   \tl_set:Nx \__stex_copymodule_module_tl {
4307     \stex_import_require_module:nnnn
4308     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4309     { \l_stex_import_path_str } { \l_stex_import_name_str }
4310   }
4311
4312   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4313     \seq_map_inline:cn {c_stex_module_##1_constants}{
4314       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #
4315       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4316         \stex_if_do_html:T {
4317           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4318             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4319               $\stex_annotate_invisible:nnn{definients}{ }\{\exp_after:wN \exp_not:N\csname l__
4320             }
4321           }
4322         }
4323         \tl_put_right:Nx \__stex_copymodule_module_tl {
4324           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4325         }
4326       }
4327     }}
4328
4329   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4330
4331   \__stex_copymodule_exec_tl
4332   \stex_if_do_html:T {\end{stex_annotate_env}}
4333 }
4334
4335 \NewDocumentCommand \donotcopy { m } {
4336   \str_clear:N \l_stex_import_name_str
4337   \str_set:Nn \l_tmpa_str { #1 }
4338   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4339   \seq_map_inline:Nn \l_stex_all_modules_seq {
4340     \str_set:Nn \l_tmpb_str { ##1 }
4341     \str_if_eq:eeT { \l_tmpa_str } {
4342       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4343     } {
4344       \seq_map_break:n {
4345         \stex_if_do_html:T {

```

```

4346         \stex_if_smsmode:F {
4347             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4348                 \stex_annotate:nnn{domain}{##1}{}}
4349         }
4350     }
4351 }
4352 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4353 }
4354 }
4355 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4356     \str_set:Nn \l_tmpb_str { #####1 }
4357     \str_if_eq:eeT { \l_tmpa_str } {
4358         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4359     } {
4360         \seq_map_break:n {\seq_map_break:n {
4361             \stex_if_do_html:T {
4362                 \stex_if_smsmode:F {
4363                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4364                         \stex_annotate:nnn{domain}{
4365                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4366                         }{}
4367                     }
4368                 }
4369             }
4370             \str_set:Nx \l_stex_import_name_str {
4371                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4372             }
4373         }}
4374     }
4375 }
4376 }
4377 \str_if_empty:NTF \l_stex_import_name_str {
4378     % TODO throw error
4379 }{
4380     \stex_collect_imports:n {\l_stex_import_name_str }
4381     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4382         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4383         \seq_map_inline:cn {c_stex_module_###1_constants}{
4384             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4385             \bool_lazy_any:nT {
4386                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4387                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4388                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4389             }{
4390                 % TODO throw error
4391             }
4392         }
4393     }
4394     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4395     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4396     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4397 }
4398 \stex_smsmode_do:
4399 }

```

```

4400
4401 \NewDocumentCommand \assign { m m }{
4402   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4403   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4404   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4405   \stex_smsmode_do:
4406 }
4407
4408 \keys_define:nn { stex / renamedekl } {
4409   name          .str_set_x:N = \l_stex_renamedekl_name_str
4410 }
4411 \cs_new_protected:Nn \__stex_copymodule_renamedekl_args:n {
4412   \str_clear:N \l_stex_renamedekl_name_str
4413   \keys_set:nn { stex / renamedekl } { #1 }
4414 }
4415
4416 \NewDocumentCommand \renamedekl { O{} m m }{
4417   \__stex_copymodule_renamedekl_args:n { #1 }
4418   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4419   \stex_debug:nn{renamedekl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4420   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4421   \str_if_empty:NTF \l_stex_renamedekl_name_str {
4422     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4423       \l_stex_get_symbol_uri_str
4424     } }
4425   } {
4426     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4427       \stex_debug:nn{renamedekl}{@~\l_stex_current_module_str ? \l_stex_renamedekl_name_str}
4428       \prop_set_eq:cc {l_stex_symdecl_
4429         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4430       _prop
4431       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4432       \seq_set_eq:cc {l_stex_symdecl_
4433         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4434       _notations
4435       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4436       \prop_put:cnx {l_stex_symdecl_
4437         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4438       _prop
4439       }{ name }{ \l_stex_renamedekl_name_str }
4440       \prop_put:cnx {l_stex_symdecl_
4441         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4442       _prop
4443       }{ module }{ \l_stex_current_module_str }
4444       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4445         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4446       }
4447       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4448         \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4449       } }
4450   }
4451   \stex_smsmode_do:
4452 }
4453

```

```

4454 \stex_deactivate_macro:Nn \assign {copymodules}
4455 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4456 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4457
4458

```

31.2 The feature environment

structural@feature

```

4459 <@@=stex_features>
4460
4461 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4462   \stex_if_in_module:F {
4463     \msg_set:nnn{stex}{error/nomodule}{
4464       Structural~Feature~has~to~occur~in~a~module:\\
4465       Feature~#2~of~type~#1\\
4466       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4467     }
4468     \msg_error:nn{stex}{error/nomodule}
4469   }
4470
4471   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4472
4473   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4474
4475   \stex_if_do_html:T {
4476     \begin{stex_annotate_env}{feature:#1}{\l_stex_feature_parent_str ? #2 - #1}
4477     \stex_annotate_invisible:nnn{header}{\l_stex_feature_parent_str ? #2 - #1}
4478   }
4479 }{
4480   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4481   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4482   \stex_debug:nn{features}{
4483     Feature: \l_stex_last_feature_str
4484   }
4485   \stex_if_do_html:T {
4486     \end{stex_annotate_env}
4487   }
4488 }

```

31.3 Structure

structure

```

4489 <@@=stex_structures>
4490 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4491   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4492     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4493   }
4494   \prop_gput:cxn {c_stex_module_ \l_stex_current_module_str _structures}
4495   {#1}{#2}
4496 }
4497

```

```

4498 \keys_define:nn { stex / features / structure } {
4499   name          .str_set_x:N = \l__stex_structures_name_str ,
4500 }
4501
4502 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4503   \str_clear:N \l__stex_structures_name_str
4504   \keys_set:nn { stex / features / structure } { #1 }
4505 }
4506
4507 \NewDocumentEnvironment{mathstructure}{m O{}}{
4508   \__stex_structures_structure_args:n { #2 }
4509   \str_if_empty:NT \l__stex_structures_name_str {
4510     \str_set:Nx \l__stex_structures_name_str { #1 }
4511   }
4512   \stex_suppress_html:n {
4513     \exp_args:Nx \stex_symdecl_do:nn {
4514       name = \l__stex_structures_name_str ,
4515       def = {\STEXsymbol{module-type}}{
4516         \stex_term_math_oms:nnnn {
4517           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4518             { ns } ?
4519           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4520             { name } / \l__stex_structures_name_str - structure
4521         }{}{0}{}
4522       }}
4523     }{ #1 }
4524   }
4525   \exp_args:Nnnx
4526   \begin{structural_feature_module}{ structure }
4527     { \l__stex_structures_name_str }{}
4528   \stex_smsmode_do:
4529 }{
4530   \end{structural_feature_module}
4531   \stex_reset_up_to_module:n \l_stex_last_feature_str
4532   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4533   \seq_clear:N \l_tmpa_seq
4534   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4535     \seq_map_inline:cn{c_stex_module_##1_constants}{
4536       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4537     }
4538   }
4539   \exp_args:Nnno
4540   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4541   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4542   \stex_add_structure_to_current_module:nn
4543     \l__stex_structures_name_str
4544     \l_stex_last_feature_str
4545
4546   \stex_execute_in_module:x {
4547     \tl_set:cn { #1 }{
4548       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4549     }
4550   }
4551 }

```

```

4552
4553 \cs_new:Nn \stex_invoke_structure:nn {
4554   \stex_invoke_symbol:n { #1?#2 }
4555 }
4556
4557 \cs_new_protected:Nn \stex_get_structure:n {
4558   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4559     \tl_set:Nn \l_tmpa_tl { #1 }
4560     \__stex_structures_get_from_cs:
4561   }{
4562     \cs_if_exist:cTF { #1 }{
4563       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4564       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4565       \str_if_empty:NNTF \l_tmpa_str {
4566         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4567           \__stex_structures_get_from_cs:
4568         }{
4569           \__stex_structures_get_from_string:n { #1 }
4570         }
4571       }{
4572         \__stex_structures_get_from_string:n { #1 }
4573       }
4574     }{
4575       \__stex_structures_get_from_string:n { #1 }
4576     }
4577   }
4578 }
4579
4580 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4581   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4582     { \tl_tail:N \l_tmpa_tl }
4583   \str_set:Nx \l_tmpa_str {
4584     \exp_after:wN \use_i:nn \l_tmpa_tl
4585   }
4586   \str_set:Nx \l_tmpb_str {
4587     \exp_after:wN \use_ii:nn \l_tmpa_tl
4588   }
4589   \str_set:Nx \l_stex_get_structure_str {
4590     \l_tmpa_str ? \l_tmpb_str
4591   }
4592   \str_set:Nx \l_stex_get_structure_module_str {
4593     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4594   }
4595 }
4596
4597 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4598   \tl_set:Nn \l_tmpa_tl {
4599     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4600   }
4601   \str_set:Nn \l_tmpa_str { #1 }
4602   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4603
4604   \seq_map_inline:Nn \l_stex_all_modules_seq {
4605     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4606 \prop_map_inline:cn {c_stex_module_##1_structures} {
4607   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4608     \prop_map_break:n{\seq_map_break:n{
4609       \tl_set:Nn \l_tmpa_tl {
4610         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4611         \str_set:Nn \l_stex_get_structure_module_str {####2}
4612       }
4613     }}
4614   }
4615 }
4616 }
4617 }
4618 \l_tmpa_tl
4619 }

```

\instantiate

```

4620
4621 \keys_define:nn { stex / instantiate } {
4622   name          .str_set_x:N = \l__stex_structures_name_str
4623 }
4624 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4625   \str_clear:N \l__stex_structures_name_str
4626   \keys_set:nn { stex / instantiate } { #1 }
4627 }
4628
4629 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4630   \begin{group}
4631     \stex_get_structure:n {#3}
4632     \__stex_structures_instantiate_args:n { #2 }
4633     \str_if_empty:NT \l__stex_structures_name_str {
4634       \str_set:Nn \l__stex_structures_name_str { #1 }
4635     }
4636     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4637     \seq_clear:N \l__stex_structures_fields_seq
4638     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4639     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4640       \seq_map_inline:cn {c_stex_module_##1_constants}{
4641         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4642       }
4643     }
4644
4645     \tl_if_empty:nF{#5}{
4646       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4647       \prop_clear:N \l_tmpa_prop
4648       \seq_map_inline:Nn \l_tmpa_seq {
4649         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4650         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4651           \msg_error:nnn{stex}{error/keyval}{##1}
4652         }
4653         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4654         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4655         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4656         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4657         \exp_args:Nxx \str_if_eq:nnF

```



```

4658         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4659         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}{
4660         \msg_error:nnxxxx{stex}{error/incompatible}
4661         {l\_stex_structures_dom_str}
4662         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4663         {\l_stex_get_symbol_uri_str}
4664         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}
4665     }
4666     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4667 }
4668 }
4669
4670 \seq_map_inline:Nn \l\_stex_structures_fields_seq {
4671     \str_set:Nx \l_tmpa_str {field:l\_stex_structures_name_str . \prop_item:cn {l_stex_sy
4672     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4673
4674     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4675     \stex_execute_in_module:x {
4676         \prop_set_from_keyval:cn { l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str_p
4677         name = \l_tmpa_str ,
4678         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4679         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4680         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4681     }
4682     \seq_clear:c {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notations}
4683 }
4684
4685 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4686     \stex_find_notation:nn{##1}{}
4687     \stex_execute_in_module:x {
4688         \seq_put_right:cn {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notation
4689     }
4690
4691     \stex_copy_control_sequence_ii:ccN
4692     {stex_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4693     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4694     \l_tmpa_tl
4695     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4696
4697
4698     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4699         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4700         \stex_execute_in_module:x {
4701             \tl_set:cn
4702             {stex_op_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4703             { \exp_args:No \exp_not:n \l_tmpa_cs}
4704         }
4705     }
4706
4707 }
4708
4709 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4710 }
4711

```

```

4712 \stex_execute_in_module:x {
4713   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4714   domain = \l_stex_get_structure_module_str ,
4715   \prop_to_keyval:N \l_tmpa_prop
4716 }
4717 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4718 }
4719 \stex_debug:nn{instantiate}{
4720   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4721   \prop_to_keyval:N \l_tmpa_prop
4722 }
4723 \exp_args:Nxx \stex_symdecl_do:nn {
4724   type={\STEXsymbol{module-type}}{
4725     \stex_term_math_oms:nnnn {
4726       \l_stex_get_structure_module_str
4727     }{}{0}{}
4728   }}
4729 }{\l__stex_structures_name_str}
4730 % {
4731   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4732   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4733   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4734 % }
4735 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4736 \endgroup
4737 \stex_smsmode_do:\ignorespacesandpars
4738 }
4739
4740 \cs_new_protected:Nn \stex_symbol_or_var:n {
4741   \cs_if_exist:cTF{#1}{
4742     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4743     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4744     \str_if_empty:NTF \l_tmpa_str {
4745       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4746       \stex_invoke_variable:n {
4747         \bool_set_true:N \l_stex_symbol_or_var_bool
4748         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4749         \str_set:Nx \l_stex_get_symbol_uri_str {
4750           \exp_after:wN \use:n \l_tmpa_tl
4751         }
4752       }{
4753         \bool_set_false:N \l_stex_symbol_or_var_bool
4754         \stex_get_symbol:n{#1}
4755       }
4756     }{
4757       \__stex_structures_symbolorvar_from_string:n{ #1 }
4758     }
4759   }{
4760     \__stex_structures_symbolorvar_from_string:n{ #1 }
4761   }
4762 }
4763
4764 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4765   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4766     \bool_set_true:N \l_stex_symbol_or_var_bool
4767     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4768   }{
4769     \bool_set_false:N \l_stex_symbol_or_var_bool
4770     \stex_get_symbol:n{#1}
4771   }
4772 }
4773
4774 \keys_define:nn { stex / varinstantiate } {
4775   name      .str_set_x:N = \l__stex_structures_name_str,
4776   bind      .choices:nn =
4777     {forall,exists}
4778     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4779 }
4780
4781 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4782   \str_clear:N \l__stex_structures_name_str
4783   \str_clear:N \l__stex_structures_bind_str
4784   \keys_set:nn { stex / varinstantiate } { #1 }
4785 }
4786
4787 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4788   \beginingroup
4789     \stex_get_structure:n {#3}
4790     \__stex_structures_varinstantiate_args:n { #2 }
4791     \str_if_empty:NT \l__stex_structures_name_str {
4792       \str_set:Nn \l__stex_structures_name_str { #1 }
4793     }
4794     \stex_if_do_html:TF{
4795       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4796     }{\use:n}
4797     {
4798       \stex_if_do_html:T{
4799         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4800       }
4801       \seq_clear:N \l__stex_structures_fields_seq
4802       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4803       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4804         \seq_map_inline:cn {c_stex_module_##1_constants}{
4805           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4806         }
4807       }
4808       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4809       \prop_clear:N \l_tmpa_prop
4810       \tl_if_empty:nF {#5} {
4811         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4812         \seq_map_inline:Nn \l_tmpa_seq {
4813           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4814           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4815             \msg_error:nnn{stex}{error/keyval}{##1}
4816           }
4817           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4818           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4819           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4820 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4821 \stex_if_do_html:T{
4822   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4823 }
4824 \bool_if:NTF \l_stex_symbol_or_var_bool {
4825   \exp_args:Nxx \str_if_eq:nnF
4826     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4827     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4828     \msg_error:nnxxxx{stex}{error/incompatible}
4829     {\l__stex_structures_dom_str
4830     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4831     {\l_stex_get_symbol_uri_str
4832     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4833   }
4834   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4835 }{
4836   \exp_args:Nxx \str_if_eq:nnF
4837     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4838     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4839     \msg_error:nnxxxx{stex}{error/incompatible}
4840     {\l__stex_structures_dom_str
4841     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4842     {\l_stex_get_symbol_uri_str
4843     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4844   }
4845   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4846 }
4847 }
4848 }
4849 \tl_gclear:N \g__stex_structures_aftergroup_tl
4850 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4851   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_
4852   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4853   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4854     \stex_find_notation:nn{##1}{
4855       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4856         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4857       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4858       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4859         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4860           {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4861         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4862       }
4863     }
4864   }
4865   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4866     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4867       name = \l_tmpa_str ,
4868       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4869       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4870       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4871     }
4872     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4873     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4874         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4875         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4876     }
4877     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4878 }
4879 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4880     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4881         domain = \l_stex_get_structure_module_str ,
4882         \prop_to_keyval:N \l_tmpa_prop
4883     }
4884     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4885     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4886         \exp_args:Nnx \exp_not:N \use:nn {
4887             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4888             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4889                 \exp_not:n{
4890                     \_varcomp{#4}
4891                 }
4892             }
4893         }{
4894             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4895         }
4896     }
4897 }
4898 }
4899 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4900 \aftergroup\g__stex_structures_aftergroup_tl
4901 \endgroup
4902 \stex_smsmode_do:\ignorespacesandpars
4903 }
4904
4905 \cs_new_protected:Nn \stex_invoke_instance:n {
4906     \peek_charcode_remove:NTF ! {
4907         \stex_invoke_symbol:n{#1}
4908     }{
4909         \_stex_invoke_instance:nn {#1}
4910     }
4911 }
4912
4913
4914 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4915     \peek_charcode_remove:NTF ! {
4916         \exp_args:Nnx \use:nn {
4917             \def\comp{\_varcomp}
4918             \use:c{l_stex_varinstance_#1_op_tl}
4919         }{
4920             \_stex_reset:N \comp
4921         }
4922     }{
4923         \_stex_invoke_varinstance:nn {#1}
4924     }
4925 }
4926
4927 \cs_new_protected:Nn \stex_invoke_instance:nn {

```

```

4928 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4929   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4930 }{
4931   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4932   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4933     \prop_to_keyval:N \l_tmpa_prop
4934   }
4935 }
4936 }
4937
4938 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4939   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4940     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4941     \l_tmpa_tl
4942   }{
4943     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4944   }
4945 }

```

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

`\stex_invoke_structure:nnn`

```

4946 % #1: URI of the instance
4947 % #2: URI of the instantiated module
4948 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4949   \tl_if_empty:nTF{ #3 }{
4950     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4951       c_stex_feature_ #2 _prop
4952     }
4953     \tl_clear:N \l_tmpa_tl
4954     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4955     \seq_map_inline:Nn \l_tmpa_seq {
4956       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4957       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4958       \cs_if_exist:cT {
4959         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4960       }{
4961         \tl_if_empty:NF \l_tmpa_tl {
4962           \tl_put_right:Nn \l_tmpa_tl {,}
4963         }
4964         \tl_put_right:Nx \l_tmpa_tl {
4965           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4966         }
4967       }
4968     }
4969     \exp_args:No \mathstrut \l_tmpa_tl
4970   }{
4971     \stex_invoke_symbol:n{#1/#3}
4972   }
4973 }

```

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

```

4974 </package>

```

Chapter 32

STEX -Statements Implementation

```
4975 <*package>
4976
4977 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4978
4979 <@@=stex_statements>
    Warnings and error messages
4980
\titleemph
4981 \def\titleemph#1{\textbf{#1}}
(End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4982 \keys_define:nn {stex / definiendum }{
4983   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4984   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4985   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4986   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4987 }
4988 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4989   \str_clear:N \l__stex_statements_definiendum_root_str
4990   \tl_clear:N \l__stex_statements_definiendum_post_tl
4991   \str_clear:N \l__stex_statements_definiendum_gfa_str
4992   \keys_set:nn { stex / definiendum }{ #1 }
4993 }
4994 \NewDocumentCommand \definiendum { O{} m m } {
4995   \__stex_statements_definiendum_args:n { #1 }
4996   \stex_get_symbol:n { #2 }
4997   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4998   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4999     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5000     \tl_set:Nn \l_tmpa_tl { #3 }
5001   } {
5002     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5003     \tl_set:Nn \l_tmpa_tl {
5004       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5005     }
5006   }
5007 } {
5008   \tl_set:Nn \l_tmpa_tl { #3 }
5009 }
5010
5011 % TODO root
5012 \stex_html_backend:TF {
5013   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5014 } {
5015   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5016 }
5017 }
5018 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

5019
5020 \NewDocumentCommand \definame { 0{ } m } {
5021   \__stex_statements_definiendum_args:n { #1 }
5022   % TODO: root
5023   \stex_get_symbol:n { #2 }
5024   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5025   \str_set:Nx \l_tmpa_str {
5026     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5027   }
5028   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5029   \stex_html_backend:TF {
5030     \stex_if_do_html:T {
5031       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5032         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5033       }
5034     }
5035   } {
5036     \exp_args:Nnx \defemph@uri {
5037       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5038     } { \l_stex_get_symbol_uri_str }
5039   }
5040 }
5041 \stex_deactivate_macro:Nn \definame {definition~environments}
5042
5043 \NewDocumentCommand \Definame { 0{ } m } {
5044   \__stex_statements_definiendum_args:n { #1 }
5045   \stex_get_symbol:n { #2 }
5046   \str_set:Nx \l_tmpa_str {
5047     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5048   }
5049   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```



```

5050 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5051 \stex_html_backend:TF {
5052   \stex_if_do_html:T {
5053     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5054       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5055     }
5056   }
5057 } {
5058   \exp_args:Nnx \defemph@uri {
5059     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5060   } { \l_stex_get_symbol_uri_str }
5061 }
5062 }
5063 \stex_deactivate_macro:Nn \Definame {definition-environments}
5064
5065 \NewDocumentCommand \premise { m }{
5066   \stex_annotate:nnn{ premise }{}{ #1 }
5067 }
5068 \NewDocumentCommand \conclusion { m }{
5069   \stex_annotate:nnn{ conclusion }{}{ #1 }
5070 }
5071 \NewDocumentCommand \definiens { 0{} m }{
5072   \str_clear:N \l_stex_get_symbol_uri_str
5073   \tl_if_empty:nF {#1} {
5074     \stex_get_symbol:n { #1 }
5075   }
5076   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5077     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5078       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5079     }{
5080       % TODO throw error
5081     }
5082   }
5083   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
5084     {\l_stex_current_module_str}{
5085       \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5086     }{true}{
5087       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5088       \exp_args:Nx \stex_add_to_current_module:n {
5089         \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5090       }
5091     }
5092   }
5093   \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5094 }
5095
5096 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5097 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5098 \stex_deactivate_macro:Nn \definiens {definition~environments}
5099

```

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

`sdefinition`

```

5100
5101 \keys_define:nn {stex / sdefinition }{
5102   type      .str_set_x:N = \sdefinitiontype,
5103   id        .str_set_x:N = \sdefinitionid,
5104   name      .str_set_x:N = \sdefinitionname,
5105   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5106   title     .tl_set:N     = \sdefinitiontitle
5107 }
5108 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5109   \str_clear:N \sdefinitiontype
5110   \str_clear:N \sdefinitionid
5111   \str_clear:N \sdefinitionname
5112   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5113   \tl_clear:N \sdefinitiontitle
5114   \keys_set:nn { stex / sdefinition }{ #1 }
5115 }
5116
5117 \NewDocumentEnvironment{sdefinition}{0{}}{
5118   \__stex_statements_sdefinition_args:n{ #1 }
5119   \stex_reactivate_macro:N \definiendum
5120   \stex_reactivate_macro:N \definame
5121   \stex_reactivate_macro:N \Definame
5122   \stex_reactivate_macro:N \premise
5123   \stex_reactivate_macro:N \definiens
5124   \stex_if_smsmode:F{
5125     \seq_clear:N \l_tmpb_seq
5126     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5127       \tl_if_empty:nF{ ##1 }{
5128         \stex_get_symbol:n { ##1 }
5129         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5130           \l_stex_get_symbol_uri_str
5131         }
5132       }
5133     }
5134     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5135     \exp_args:Nnnx
5136     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5137     \str_if_empty:NF \sdefinitiontype {
5138       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5139     }
5140     \str_if_empty:NF \sdefinitionname {
5141       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5142     }
5143     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5144     \tl_clear:N \l_tmpa_tl
5145     \clist_map_inline:Nn \l_tmpa_clist {
5146       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5147         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5148       }
5149     }
5150     \tl_if_empty:NTF \l_tmpa_tl {
5151       \__stex_statements_sdefinition_start:
5152     }{
5153       \l_tmpa_tl

```

```

5154     }
5155   }
5156   \stex_ref_new_doc_target:n \sdefinitionid
5157   \stex_smsmode_do:
5158 }{
5159   \stex_suppress_html:n {
5160     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5161   }
5162   \stex_if_smsmode:F {
5163     \clist_set:No \l_tmpa_clist \sdefinitiontype
5164     \tl_clear:N \l_tmpa_tl
5165     \clist_map_inline:Nn \l_tmpa_clist {
5166       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5167         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5168       }
5169     }
5170     \tl_if_empty:NTF \l_tmpa_tl {
5171       \__stex_statements_sdefinition_end:
5172     }{
5173       \l_tmpa_tl
5174     }
5175     \end{stex_annotate_env}
5176   }
5177 }

```

\stexpatchdefinition

```

5178 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5179   \stex_par:\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5180     ~(\sdefinitiontitle)
5181   }~}
5182 }
5183 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5184
5185 \newcommand\stexpatchdefinition[3] [] {
5186   \str_set:Nx \l_tmpa_str{ #1 }
5187   \str_if_empty:NTF \l_tmpa_str {
5188     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5189     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5190   }{
5191     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5192     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5193   }
5194 }

```

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

\inlinedef inline:

```

5195 \keys_define:nn {stex / inlinedef }{
5196   type      .str_set_x:N = \sdefinitiontype,
5197   id        .str_set_x:N = \sdefinitionid,
5198   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5199   name      .str_set_x:N = \sdefinitionname
5200 }
5201 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5202 \str_clear:N \sdefinitiontype
5203 \str_clear:N \sdefinitionid
5204 \str_clear:N \sdefinitionname
5205 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5206 \keys_set:nn { stex / inlinedef }{ #1 }
5207 }
5208 \NewDocumentCommand \inlinedef { 0{} m } {
5209   \begingroup
5210   \__stex_statements_inlinedef_args:n{ #1 }
5211   \stex_reactivate_macro:N \definiendum
5212   \stex_reactivate_macro:N \definame
5213   \stex_reactivate_macro:N \Definame
5214   \stex_reactivate_macro:N \premise
5215   \stex_reactivate_macro:N \definiens
5216   \stex_ref_new_doc_target:n \sdefinitionid
5217   \stex_if_smsmode:TF{\stex_suppress_html:n {
5218     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5219   }}{
5220     \seq_clear:N \l_tmpb_seq
5221     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5222       \tl_if_empty:nF{ ##1 }{
5223         \stex_get_symbol:n { ##1 }
5224         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5225           \l_stex_get_symbol_uri_str
5226         }
5227       }
5228     }
5229     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5230     \exp_args:Nnx
5231     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5232       \str_if_empty:NF \sdefinitiontype {
5233         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5234       }
5235       #2
5236       \str_if_empty:NF \sdefinitionname {
5237         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5238         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5239       }
5240     }
5241   }
5242   \endgroup
5243   \stex_smsmode_do:
5244 }

```

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

32.2 Assertions

sassertion

```

5245
5246 \keys_define:nn {stex / sassertion }{
5247   type      .str_set_x:N = \sassertiontype,
5248   id        .str_set_x:N = \sassertionid,

```

```

5249 title .tl_set:N = \sassertiontitle ,
5250 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5251 name .str_set_x:N = \sassertionname
5252 }
5253 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
5254 \str_clear:N \sassertiontype
5255 \str_clear:N \sassertionid
5256 \str_clear:N \sassertionname
5257 \clist_clear:N \l__stex_statements_sassertion_for_clist
5258 \tl_clear:N \sassertiontitle
5259 \keys_set:nn { stex / sassertion }{ #1 }
5260 }
5261
5262 %\tl_new:N \g__stex_statements_aftergroup_tl
5263
5264 \NewDocumentEnvironment{sassertion}{0{}}{
5265 \l__stex_statements_sassertion_args:n{ #1 }
5266 \stex_reactivate_macro:N \premise
5267 \stex_reactivate_macro:N \conclusion
5268 \stex_if_smsmode:F {
5269 \seq_clear:N \l_tmpb_seq
5270 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5271 \tl_if_empty:nF{ ##1 }{
5272 \stex_get_symbol:n { ##1 }
5273 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5274 \l_stex_get_symbol_uri_str
5275 }
5276 }
5277 }
5278 \exp_args:Nnnx
5279 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
5280 \str_if_empty:NF \sassertiontype {
5281 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5282 }
5283 \str_if_empty:NF \sassertionname {
5284 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5285 }
5286 \clist_set:Nn \l_tmpa_clist \sassertiontype
5287 \tl_clear:N \l_tmpa_tl
5288 \clist_map_inline:Nn \l_tmpa_clist {
5289 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5290 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5291 }
5292 }
5293 \tl_if_empty:NTF \l_tmpa_tl {
5294 \l__stex_statements_sassertion_start:
5295 }{
5296 \l_tmpa_tl
5297 }
5298 }
5299 \str_if_empty:NTF \sassertionid {
5300 \str_if_empty:NF \sassertionname {
5301 \stex_ref_new_doc_target:n {}
5302 }

```

```

5303 } {
5304   \stex_ref_new_doc_target:n \sassertionid
5305 }
5306 \stex_smsmode_do:
5307 ){
5308   \str_if_empty:NF \sassertionname {
5309     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5310     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5311   }
5312   \stex_if_smsmode:F {
5313     \clist_set:Nn \l_tmpa_clist \sassertiontype
5314     \tl_clear:N \l_tmpa_tl
5315     \clist_map_inline:Nn \l_tmpa_clist {
5316       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5317         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5318       }
5319     }
5320     \tl_if_empty:NTF \l_tmpa_tl {
5321       __stex_statements_sassertion_end:
5322     }{
5323       \l_tmpa_tl
5324     }
5325     \end{stex_annotate_env}
5326   }
5327 }

```

\stexpatchassertion

```

5328
5329 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5330   \stex_par:\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5331     (\sassertiontitle)
5332   }~}
5333 }
5334 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5335
5336 \newcommand\stexpatchassertion[3] [] {
5337   \str_set:Nx \l_tmpa_str{ #1 }
5338   \str_if_empty:NTF \l_tmpa_str {
5339     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5340     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5341   }{
5342     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5343     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5344   }
5345 }

```

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

\inlineass inline:

```

5346 \keys_define:nn {stex / inlineass }{
5347   type      .str_set_x:N = \sassertiontype,
5348   id        .str_set_x:N = \sassertionid,
5349   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5350   name      .str_set_x:N = \sassertionname

```

```

5351 }
5352 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5353   \str_clear:N \sassertiontype
5354   \str_clear:N \sassertionid
5355   \str_clear:N \sassertionname
5356   \clist_clear:N \l__stex_statements_sassertion_for_clist
5357   \keys_set:nn { stex / inlineass }{ #1 }
5358 }
5359 \NewDocumentCommand \inlineass { 0{} m } {
5360   \begingroup
5361     \stex_reactivate_macro:N \premise
5362     \stex_reactivate_macro:N \conclusion
5363     \__stex_statements_inlineass_args:n{ #1 }
5364     \str_if_empty:NTF \sassertionid {
5365       \str_if_empty:NF \sassertionname {
5366         \stex_ref_new_doc_target:n {}
5367       }
5368     } {
5369       \stex_ref_new_doc_target:n \sassertionid
5370     }
5371
5372     \stex_if_smsmode:TF{
5373       \str_if_empty:NF \sassertionname {
5374         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5375         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5376       }
5377     }{
5378       \seq_clear:N \l_tmpb_seq
5379       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5380         \tl_if_empty:nF{ ##1 }{
5381           \stex_get_symbol:n { ##1 }
5382           \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5383             \l_stex_get_symbol_uri_str
5384           }
5385         }
5386       }
5387       \exp_args:Nnx
5388       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {,}}{
5389         \str_if_empty:NF \sassertiontype {
5390           \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5391         }
5392         #2
5393         \str_if_empty:NF \sassertionname {
5394           \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5395           \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5396           \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5397         }
5398       }
5399     }
5400   \endgroup
5401   \stex_smsmode_do:
5402 }

```

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

32.3 Examples

sexample

```

5403
5404 \keys_define:nn {stex / sexample }{
5405   type      .str_set_x:N = \exampletype,
5406   id        .str_set_x:N = \sexampleid,
5407   title     .tl_set:N     = \sexampletitle,
5408   name      .str_set_x:N = \sexamplename ,
5409   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5410 }
5411 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5412   \str_clear:N \sexampletype
5413   \str_clear:N \sexampleid
5414   \str_clear:N \sexamplename
5415   \tl_clear:N \sexampletitle
5416   \clist_clear:N \l__stex_statements_sexample_for_clist
5417   \keys_set:nn { stex / sexample }{ #1 }
5418 }
5419
5420 \NewDocumentEnvironment{sexample}{0{}}{
5421   \__stex_statements_sexample_args:n{ #1 }
5422   \stex_reactivate_macro:N \premise
5423   \stex_reactivate_macro:N \conclusion
5424   \stex_if_smsmode:F {
5425     \seq_clear:N \l_tmpb_seq
5426     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5427       \tl_if_empty:nF{ ##1 }{
5428         \stex_get_symbol:n { ##1 }
5429         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5430           \l_stex_get_symbol_uri_str
5431         }
5432       }
5433     }
5434     \exp_args:Nnnx
5435     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
5436     \str_if_empty:NF \sexampletype {
5437       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5438     }
5439     \str_if_empty:NF \sexamplename {
5440       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5441     }
5442     \clist_set:Nn \l_tmpa_clist \sexampletype
5443     \tl_clear:N \l_tmpa_tl
5444     \clist_map_inline:Nn \l_tmpa_clist {
5445       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5446         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5447       }
5448     }
5449     \tl_if_empty:NTF \l_tmpa_tl {
5450       \__stex_statements_sexample_start:
5451     }{
5452       \l_tmpa_tl
5453     }

```



```

5454 }
5455 \str_if_empty:NF \sexampleid {
5456   \stex_ref_new_doc_target:n \sexampleid
5457 }
5458 \stex_smsmode_do:
5459 ){
5460   \str_if_empty:NF \sexamplename {
5461     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5462   }
5463   \stex_if_smsmode:F {
5464     \clist_set:Nn \l_tmpa_clist \sexamplotype
5465     \tl_clear:N \l_tmpa_tl
5466     \clist_map_inline:Nn \l_tmpa_clist {
5467       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5468         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5469       }
5470     }
5471     \tl_if_empty:NTF \l_tmpa_tl {
5472       \__stex_statements_sexample_end:
5473     }{
5474       \l_tmpa_tl
5475     }
5476     \end{stex_annotate_env}
5477   }
5478 }

```

\stexpatchexample

```

5479
5480 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5481   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexamplitle {
5482     (\sexamplitle)
5483   }~}
5484 }
5485 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5486
5487 \newcommand\stexpatchexample[3]{} {
5488   \str_set:Nx \l_tmpa_str{ #1 }
5489   \str_if_empty:NTF \l_tmpa_str {
5490     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5491     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5492   }{
5493     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5494     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5495   }
5496 }

```

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

\inlineex inline:

```

5497 \keys_define:nn {stex / inlineex }{
5498   type      .str_set_x:N = \sexamplotype,
5499   id        .str_set_x:N = \sexampleid,
5500   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5501   name      .str_set_x:N = \sexamplename

```

```

5502 }
5503 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5504   \str_clear:N \sexamplotype
5505   \str_clear:N \sexampleid
5506   \str_clear:N \sexamplename
5507   \clist_clear:N \l__stex_statements_sexample_for_clist
5508   \keys_set:nn { stex / inlineex }{ #1 }
5509 }
5510 \NewDocumentCommand \inlineex { 0{ } m } {
5511   \beginngroup
5512   \stex_reactivate_macro:N \premise
5513   \stex_reactivate_macro:N \conclusion
5514   \__stex_statements_inlineex_args:n{ #1 }
5515   \str_if_empty:NF \sexampleid {
5516     \stex_ref_new_doc_target:n \sexampleid
5517   }
5518   \stex_if_smsmode:TF{
5519     \str_if_empty:NF \sexamplename {
5520       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5521     }
5522   }{
5523     \seq_clear:N \l_tmpb_seq
5524     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5525       \tl_if_empty:nF{ ##1 }{
5526         \stex_get_symbol:n { ##1 }
5527         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5528           \l_stex_get_symbol_uri_str
5529         }
5530       }
5531     }
5532     \exp_args:Nnx
5533     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {},}}{
5534       \str_if_empty:NF \sexamplotype {
5535         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5536       }
5537       #2
5538       \str_if_empty:NF \sexamplename {
5539         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5540         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5541       }
5542     }
5543   }
5544   \endgroup
5545   \stex_smsmode_do:
5546 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5547 \keys_define:nn { stex / sparagraph } {
5548   id          .str_set_x:N    = \sparagraphid ,

```

```

5549 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5550 type .str_set_x:N = \sparagraphtype ,
5551 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5552 from .tl_set:N = \sparagraphfrom ,
5553 to .tl_set:N = \sparagraphto ,
5554 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5555 name .str_set:N = \sparagraphname ,
5556 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5557 }
5558
5559 \cs_new_protected:Nn \stex_sparagraph_args:n {
5560 \tl_clear:N \l_stex_sparagraph_title_tl
5561 \tl_clear:N \sparagraphfrom
5562 \tl_clear:N \sparagraphto
5563 \tl_clear:N \l_stex_sparagraph_start_tl
5564 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5565 \str_clear:N \sparagraphid
5566 \str_clear:N \sparagraphtype
5567 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5568 \str_clear:N \sparagraphname
5569 \keys_set:nn { stex / sparagraph }{ #1 }
5570 }
5571 \newif\if@in@omtext\@in@omtextfalse
5572
5573 \NewDocumentEnvironment {sparagraph} { 0{} } {
5574 \stex_sparagraph_args:n { #1 }
5575 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5576 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5577 }{
5578 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5579 }
5580 \@in@omtexttrue
5581 \stex_if_smsmode:F {
5582 \seq_clear:N \l_tmpb_seq
5583 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5584 \tl_if_empty:NF{ ##1 }{
5585 \stex_get_symbol:n { ##1 }
5586 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5587 \l_stex_get_symbol_uri_str
5588 }
5589 }
5590 }
5591 \exp_args:Nnnx
5592 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5593 \str_if_empty:NF \sparagraphtype {
5594 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5595 }
5596 \str_if_empty:NF \sparagraphfrom {
5597 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5598 }
5599 \str_if_empty:NF \sparagraphto {
5600 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5601 }
5602 \str_if_empty:NF \sparagraphname {

```

```

5603     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5604   }
5605   \clist_set:No \l_tmpa_clist \sparagraphtype
5606   \tl_clear:N \l_tmpa_tl
5607   \clist_map_inline:Nn \sparagraphtype {
5608     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5609       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5610     }
5611   }
5612   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5613   \tl_if_empty:NTF \l_tmpa_tl {
5614     \__stex_statements_sparagraph_start:
5615   }{
5616     \l_tmpa_tl
5617   }
5618 }
5619 \clist_set:No \l_tmpa_clist \sparagraphtype
5620 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5621 {
5622   \stex_reactivate_macro:N \definiendum
5623   \stex_reactivate_macro:N \definame
5624   \stex_reactivate_macro:N \Definame
5625   \stex_reactivate_macro:N \premise
5626   \stex_reactivate_macro:N \definiens
5627 }
5628 \str_if_empty:NTF \sparagraphid {
5629   \str_if_empty:NTF \sparagraphname {
5630     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5631       \stex_ref_new_doc_target:n {}
5632     }
5633   } {
5634     \stex_ref_new_doc_target:n {}
5635   }
5636 } {
5637   \stex_ref_new_doc_target:n \sparagraphid
5638 }
5639 \exp_args:NNx
5640 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5641   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5642     \tl_if_empty:nF{ ##1 }{
5643       \stex_get_symbol:n { ##1 }
5644       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5645     }
5646   }
5647 }
5648 \stex_smsmode_do:
5649 \ignorespacesandpars
5650 }{
5651   \str_if_empty:NF \sparagraphname {
5652     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5653     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5654   }
5655   \stex_if_smsmode:F {
5656     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5657 \tl_clear:N \l_tmpa_tl
5658 \clist_map_inline:Nn \l_tmpa_clist {
5659   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5660     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5661   }
5662 }
5663 \tl_if_empty:NTF \l_tmpa_tl {
5664   \__stex_statements_sparagraph_end:
5665 }{
5666   \l_tmpa_tl
5667 }
5668 \end{stex_annotate_env}
5669 }
5670 }

```

\stexpatchparagraph

```

5671
5672 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5673   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5674     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5675       \titleemph{\l_stex_sparagraph_title_tl}:~
5676     }
5677   }{
5678     \titleemph{\l_stex_sparagraph_start_tl}~
5679   }
5680 }
5681 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5682
5683 \newcommand\stexpatchparagraph[3] [] {
5684   \str_set:Nx \l_tmpa_str{ #1 }
5685   \str_if_empty:NTF \l_tmpa_str {
5686     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5687     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5688   }{
5689     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5690     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5691   }
5692 }
5693
5694 \keys_define:nn { stex / inlinepara} {
5695   id      .str_set_x:N = \sparagraphid ,
5696   type    .str_set_x:N = \sparagraphtype ,
5697   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5698   from    .tl_set:N    = \sparagraphfrom ,
5699   to      .tl_set:N    = \sparagraphto ,
5700   name    .str_set:N   = \sparagraphname
5701 }
5702 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5703   \tl_clear:N \sparagraphfrom
5704   \tl_clear:N \sparagraphto
5705   \str_clear:N \sparagraphid
5706   \str_clear:N \sparagraphtype
5707   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5708   \str_clear:N \sparagraphname

```

```

5709 \keys_set:nn { stex / inlinepara }{ #1 }
5710 }
5711 \NewDocumentCommand \inlinepara { 0{} m } {
5712   \beginingroup
5713   \__stex_statements_inlinepara_args:n{ #1 }
5714   \clist_set:No \l_tmpa_clist \sparagraphtype
5715   \str_if_empty:NTF \sparagraphid {
5716     \str_if_empty:NTF \sparagraphname {
5717       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5718         \stex_ref_new_doc_target:n {}
5719       }
5720     } {
5721       \stex_ref_new_doc_target:n {}
5722     }
5723   } {
5724     \stex_ref_new_doc_target:n \sparagraphid
5725   }
5726   \stex_if_smsmode:TF{
5727     \str_if_empty:NF \sparagraphname {
5728       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5729     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5730   }
5731 }{
5732   \seq_clear:N \l_tmpb_seq
5733   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5734     \tl_if_empty:nF{ ##1 }{
5735       \stex_get_symbol:n { ##1 }
5736       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5737         \l_stex_get_symbol_uri_str
5738       }
5739     }
5740   }
5741   \exp_args:Nnx
5742   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5743     \str_if_empty:NF \sparagraphtype {
5744       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5745     }
5746     \str_if_empty:NF \sparagraphfrom {
5747       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5748     }
5749     \str_if_empty:NF \sparagraphto {
5750       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5751     }
5752     \str_if_empty:NF \sparagraphname {
5753       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5754     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5755     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5756   }
5757   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5758     \clist_map_inline:Nn \l_tmpb_seq {
5759       \stex_ref_new_sym_target:n {##1}
5760     }
5761   }
5762   #2

```

```

5763     }
5764   }
5765   \endgroup
5766   \stex_smsmode_do:
5767 }
5768

```

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

```

5769 \endpackage

```

Chapter 33

The Implementation

```
5770 <*package>
5771 <@@=stex_sproof>
5772
5773 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5774
```

33.1 Proofs

We first define some keys for the proof environment.

```
5775 \keys_define:nn { stex / spf } {
5776   id          .str_set_x:N = \spfid,
5777   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5778   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5779   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5780   type        .str_set_x:N = \spftype,
5781   title       .tl_set:N    = \spftitle,
5782   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5783   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5784   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5785 }
5786 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5787   \str_clear:N \spfid
5788   \tl_clear:N \l__stex_sproof_spf_for_tl
5789   \tl_clear:N \l__stex_sproof_spf_from_tl
5790   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5791   \str_clear:N \spftype
5792   \tl_clear:N \spftitle
5793   \tl_clear:N \l__stex_sproof_spf_continues_tl
5794   \tl_clear:N \l__stex_sproof_spf_functions_tl
5795   \tl_clear:N \l__stex_sproof_spf_method_tl
5796   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5797   \keys_set:nn { stex / spf }{ #1 }
5798 }
```

\c__stex_sproof_flow_str

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

```
5799 \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.

```

5800 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5801 \cs_new_protected:Npn \sproofnumber {
5802   \int_set:Nn \l_tmpa_int {1}
5803   \bool_while_do:nn {
5804     \int_compare_p:nNn {
5805       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5806     } > 0
5807   }{
5808     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5809     \int_incr:N \l_tmpa_int
5810   }
5811 }
5812 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5813   \int_set:Nn \l_tmpa_int {1}
5814   \bool_while_do:nn {
5815     \int_compare_p:nNn {
5816       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5817     } > 0
5818   }{
5819     \int_incr:N \l_tmpa_int
5820   }
5821   \int_compare:nNnF \l_tmpa_int = 1 {
5822     \int_decr:N \l_tmpa_int
5823   }
5824   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5825     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5826   }
5827 }
5828
5829 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5830   \int_set:Nn \l_tmpa_int {1}
5831   \bool_while_do:nn {
5832     \int_compare_p:nNn {
5833       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5834     } > 0
5835   }{
5836     \int_incr:N \l_tmpa_int
5837   }
5838   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5839 }
5840
5841 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5842   \int_set:Nn \l_tmpa_int {1}
5843   \bool_while_do:nn {

```

```

5844 \int_compare_p:nNn {
5845 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5846 } > 0
5847 }{
5848 \int_incr:N \l_tmpa_int
5849 }
5850 \int_decr:N \l_tmpa_int
5851 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5852 }

```

\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

```

5853 \def\sproof@box{
5854 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5855 }
5856 \def\sproofend{
5857 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5858 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5859 }
5860 }

```

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

spf@*@kw

```

5861 \def\spf@proofsketch@kw{Proof~Sketch}
5862 \def\spf@proof@kw{Proof}
5863 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5864 \AddToHook{begindocument}{
5865 \ltx@ifpackageloaded{babel}{
5866 \makeatletter
5867 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5868 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5869 \input{sproof-ngerman.ldf}
5870 }
5871 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5872 \input{sproof-finnish.ldf}
5873 }
5874 \clist_if_in:NnT \l_tmpa_clist {french}{
5875 \input{sproof-french.ldf}
5876 }
5877 \clist_if_in:NnT \l_tmpa_clist {russian}{
5878 \input{sproof-russian.ldf}
5879 }
5880 \makeatother
5881 }{}
5882 }

```

spfsketch

```

5883 \newcommand\spsketch[2][]{
5884 \begin{group}
5885 \let \premise \stex_proof_premise:

```

```

5886 \__stex_sproof_spf_args:n{#1}
5887 \stex_if_smsmode:TF {
5888   \str_if_empty:NF \spfid {
5889     \stex_ref_new_doc_target:n \spfid
5890   }
5891 }{
5892   \seq_clear:N \l_tmpa_seq
5893   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5894     \tl_if_empty:nF{ ##1 }{
5895       \stex_get_symbol:n { ##1 }
5896       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5897         \l_stex_get_symbol_uri_str
5898       }
5899     }
5900   }
5901   \exp_args:Nnx
5902   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5903     \str_if_empty:NF \spftype {
5904       \stex_annotate_invisible:nnn{type}{\spftype}{
5905     }
5906     \clist_set:No \l_tmpa_clist \spftype
5907     \tl_set:Nn \l_tmpa_tl {
5908       \titleemph{
5909         \tl_if_empty:NTF \spftitle {
5910           \spf@proofsketch@kw
5911         }{
5912           \spftitle
5913         }
5914       }::~
5915     }
5916     \clist_map_inline:Nn \l_tmpa_clist {
5917       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5918         \tl_clear:N \l_tmpa_tl
5919       }
5920     }
5921     \str_if_empty:NF \spfid {
5922       \stex_ref_new_doc_target:n \spfid
5923     }
5924     \l_tmpa_tl #2 \sproofend
5925   }
5926 }
5927 \endgroup
5928 \stex_smsmode_do:
5929 }
5930

```

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

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

```

5931 \newenvironment{spfeq}[2][ ]{
5932   \__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

```

5933 \let \premise \stex_proof_premise:
5934 \stex_if_smsmode:TF {
5935   \str_if_empty:NF \spfid {
5936     \stex_ref_new_doc_target:n \spfid
5937   }
5938 }{
5939   \seq_clear:N \l_tmpa_seq
5940   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5941     \tl_if_empty:nF{ ##1 }{
5942       \stex_get_symbol:n { ##1 }
5943       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5944         \l_stex_get_symbol_uri_str
5945       }
5946     }
5947   }
5948   \exp_args:Nnnx
5949   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5950   \str_if_empty:NF \spftype {
5951     \stex_annotate_invisible:nnn{type}{\spftype}{ }
5952   }
5953
5954   \clist_set:No \l_tmpa_clist \spftype
5955   \tl_clear:N \l_tmpa_tl
5956   \clist_map_inline:Nn \l_tmpa_clist {
5957     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5958       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5959     }
5960     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5961       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5962     }
5963   }
5964   \tl_if_empty:NTF \l_tmpa_tl {
5965     \__stex_sproof_spfeq_start:
5966   }{
5967     \l_tmpa_tl
5968   }{~#2}
5969   \str_if_empty:NF \spfid {
5970     \stex_ref_new_doc_target:n \spfid
5971   }
5972   \begin{displaymath}\begin{array}{rcll}
5973   }
5974   \stex_smsmode_do:
5975 }{
5976   \stex_if_smsmode:F {
5977     \end{array}\end{displaymath}
5978     \clist_set:No \l_tmpa_clist \spftype
5979     \tl_clear:N \l_tmpa_tl
5980     \clist_map_inline:Nn \l_tmpa_clist {
5981       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5982         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5983       }
5984     }
5985     \tl_if_empty:NTF \l_tmpa_tl {
5986       \__stex_sproof_spfeq_end:

```

```

5987   }{
5988     \l_tmpa_tl
5989   }
5990   \end{stex_annotate_env}
5991 }
5992 }
5993
5994 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5995   \titleemph{
5996     \tl_if_empty:NTF \spftitle {
5997       \spf@proof@kw
5998     }{
5999       \spftitle
6000     }
6001   }:
6002 }
6003 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6004
6005 \newcommand\stexpatchspfeq[3] [] {
6006   \str_set:Nx \l_tmpa_str{ #1 }
6007   \str_if_empty:NTF \l_tmpa_str {
6008     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6009     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6010   }{
6011     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6012     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6013   }
6014 }
6015

```

(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.

```

6016 \newenvironment{sproof}[2] []{
6017   \let \premise \stex_proof_premise:
6018   \intarray_gzero:N \l__stex_sproof_counter_intarray
6019   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
6020   \__stex_sproof_spf_args:n{#1}
6021   \stex_if_smsmode:TF {
6022     \str_if_empty:NF \spfid {
6023       \stex_ref_new_doc_target:n \spfid
6024     }
6025   }{
6026     \seq_clear:N \l_tmpa_seq
6027     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6028       \tl_if_empty:nF{ ##1 }{
6029         \stex_get_symbol:n { ##1 }
6030         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6031           \l_stex_get_symbol_uri_str
6032         }
6033       }
6034     }

```

```

6035 \exp_args:Nnnx
6036 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6037 \str_if_empty:NF \spftype {
6038   \stex_annotate_invisible:nnn{type}{\spftype}{}}
6039 }
6040
6041 \clist_set:No \l_tmpa_clist \spftype
6042 \tl_clear:N \l_tmpa_tl
6043 \clist_map_inline:Nn \l_tmpa_clist {
6044   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6045     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6046   }
6047   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6048     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6049   }
6050 }
6051 \tl_if_empty:NTF \l_tmpa_tl {
6052   \__stex_sproof_sproof_start:
6053 }{
6054   \l_tmpa_tl
6055 }{~#2}
6056 \str_if_empty:NF \spfid {
6057   \stex_ref_new_doc_target:n \spfid
6058 }
6059 \begin{description}
6060 }
6061 \stex_smsmode_do:
6062 }{
6063   \stex_if_smsmode:F{
6064     \end{description}
6065     \clist_set:No \l_tmpa_clist \spftype
6066     \tl_clear:N \l_tmpa_tl
6067     \clist_map_inline:Nn \l_tmpa_clist {
6068       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6069         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6070       }
6071     }
6072     \tl_if_empty:NTF \l_tmpa_tl {
6073       \__stex_sproof_sproof_end:
6074     }{
6075       \l_tmpa_tl
6076     }
6077     \end{stex_annotate_env}
6078   }
6079 }
6080
6081 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6082   \par\noindent\titleemph{
6083     \tl_if_empty:NTF \spftype {
6084       \spf@proof@kw
6085     }{
6086       \spftype
6087     }
6088   }::

```

```

6089 }
6090 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6091
6092 \newcommand\stexpatchproof[3] [] {
6093   \str_set:Nx \l_tmpa_str{ #1 }
6094   \str_if_empty:NTF \l_tmpa_str {
6095     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6096     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6097   }{
6098     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6099     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6100   }
6101 }

```

\spfidea

```

6102 \newcommand\spfidea[2] []{
6103   \__stex_sproof_spf_args:n{#1}
6104   \titleemph{
6105     \tl_if_empty:NTF \spftype {Proof-Idea}{
6106       \spftype
6107     }:
6108   }~#2
6109   \sproofend
6110 }

```

(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

```

6111 \newenvironment{spfstep}[1] []{
6112   \__stex_sproof_spf_args:n{#1}
6113   \stex_if_smsmode:TF {
6114     \str_if_empty:NF \spfid {
6115       \stex_ref_new_doc_target:n \spfid
6116     }
6117   }{
6118     \@in@omtexttrue
6119     \seq_clear:N \l_tmpa_seq
6120     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6121       \tl_if_empty:nF{ ##1 }{
6122         \stex_get_symbol:n { ##1 }
6123         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6124           \l_stex_get_symbol_uri_str
6125         }
6126       }
6127     }
6128     \exp_args:Nnnx
6129     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6130     \str_if_empty:NF \spftype {
6131       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6132   }
6133   \clist_set:No \l_tmpa_clist \spftype
6134   \tl_set:Nn \l_tmpa_tl {
6135     \item[\sproofnumber]
6136     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6137   }
6138   \clist_map_inline:Nn \l_tmpa_clist {
6139     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6140       \tl_clear:N \l_tmpa_tl
6141     }
6142   }
6143   \l_tmpa_tl
6144   \tl_if_empty:NF \spftitle {
6145     {(\titleemph{\spftitle})\enspace}
6146   }
6147   \str_if_empty:NF \spfid {
6148     \stex_ref_new_doc_target:n \spfid
6149   }
6150 }
6151 \stex_smsmode_do:
6152 \ignorespacesandpars
6153 }{
6154   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6155     \__stex_sproof_inc_counter:
6156   }
6157   \stex_if_smsmode:F {
6158     \end{stex_annotate_env}
6159   }
6160 }

```

spfcomment

```

6161 \newenvironment{spfcomment}[1][]{
6162   \__stex_sproof_spf_args:n{#1}
6163   \clist_set:No \l_tmpa_clist \spftype
6164   \tl_set:Nn \l_tmpa_tl {
6165     \item[\sproofnumber]
6166     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6167   }
6168   \clist_map_inline:Nn \l_tmpa_clist {
6169     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6170       \tl_clear:N \l_tmpa_tl
6171     }
6172   }
6173   \l_tmpa_tl
6174 }{
6175   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6176     \__stex_sproof_inc_counter:
6177   }
6178 }

```

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.


```

6179 \newenvironment{subproof}[2][]{
6180   \_stex_sproof_spf_args:n{#1}
6181   \stex_if_smsmode:TF{
6182     \str_if_empty:NF \spfid {
6183       \stex_ref_new_doc_target:n \spfid
6184     }
6185   }{
6186     \seq_clear:N \l_tmpa_seq
6187     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6188       \tl_if_empty:nF{ ##1 }{
6189         \stex_get_symbol:n { ##1 }
6190         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6191           \l_stex_get_symbol_uri_str
6192         }
6193       }
6194     }
6195     \exp_args:Nnnx
6196     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6197     \str_if_empty:NF \spftype {
6198       \stex_annotate_invisible:nnn{type}{\spftype}{}
6199     }
6200
6201     \clist_set:No \l_tmpa_clist \spftype
6202     \tl_set:Nn \l_tmpa_tl {
6203       \item[\sproofnumber]
6204       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6205     }
6206     \clist_map_inline:Nn \l_tmpa_clist {
6207       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6208         \tl_clear:N \l_tmpa_tl
6209       }
6210     }
6211     \l_tmpa_tl
6212     \tl_if_empty:NF \spftitle {
6213       {(\titleemph{\spftitle})\enspace}
6214     }
6215     {~#2}
6216     \str_if_empty:NF \spfid {
6217       \stex_ref_new_doc_target:n \spfid
6218     }
6219   }
6220   \_stex_sproof_add_counter:
6221   \stex_smsmode_do:
6222 }{
6223   \_stex_sproof_remove_counter:
6224   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6225     \_stex_sproof_inc_counter:
6226   }
6227   \stex_if_smsmode:F{
6228     \end{stex_annotate_env}
6229   }
6230 }

```

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

```

6231 \newenvironment{spfcases}[2] [] {
6232   \tl_if_empty:nTF{#1}{
6233     \begin{subproof}[method=by-cases]{#2}
6234   }{
6235     \begin{subproof}[#1,method=by-cases]{#2}
6236   }
6237 }{
6238   \end{subproof}
6239 }

```

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

```

6240 \newenvironment{spfcase}[2] [] {
6241   \__stex_sproof_spf_args:n{#1}
6242   \stex_if_smsmode:TF {
6243     \str_if_empty:NF \spfid {
6244       \stex_ref_new_doc_target:n \spfid
6245     }
6246   }{
6247     \seq_clear:N \l_tmpa_seq
6248     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6249       \tl_if_empty:nF{ ##1 }{
6250         \stex_get_symbol:n { ##1 }
6251         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6252           \l_stex_get_symbol_uri_str
6253         }
6254       }
6255     }
6256     \exp_args:Nnnx
6257     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6258     \str_if_empty:NF \spftype {
6259       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6260     }
6261     \clist_set:No \l_tmpa_clist \spftype
6262     \tl_set:Nn \l_tmpa_tl {
6263       \item[\sproofnumber]
6264       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6265     }
6266     \clist_map_inline:Nn \l_tmpa_clist {
6267       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6268         \tl_clear:N \l_tmpa_tl
6269       }
6270     }
6271     \l_tmpa_tl
6272     \tl_if_empty:nF{#2}{
6273       \titleemph{#2}:~
6274     }
6275   }
6276   \__stex_sproof_add_counter:
6277   \stex_smsmode_do:
6278 }{
6279   \__stex_sproof_remove_counter:
6280   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6281     \__stex_sproof_inc_counter:

```

```

6282 }
6283 \stex_if_smsmode:F{
6284   \clist_set:No \l_tmpa_clist \spftype
6285   \tl_set:Nn \l_tmpa_tl{\sproofend}
6286   \clist_map_inline:Nn \l_tmpa_clist {
6287     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6288       \tl_clear:N \l_tmpa_tl
6289     }
6290   }
6291   \l_tmpa_tl
6292   \end{stex_annotate_env}
6293 }
6294 }

```

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

```

6295 \newcommand\spfcasesketch[3] [] {
6296   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6297 }

```

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.

```

6298 \keys_define:nn { stex / just }{
6299   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6300   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6301   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6302   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6303 }

```

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

\spfjust

```

6304 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6305 \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.

```

6306 \newcommand\justarg[2] [] {#2}
6307 \</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

```
6308 <*package>
6309
6310 %%%%%%%%%% others.dtx %%%%%%%%%%
6311
6312 <@@=stex_others>
        Warnings and error messages
6313 % None

\MSC Math subject classifier

6314 \NewDocumentCommand \MSC {m} {
6315 % TODO
6316 }

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

6317 \@ifpackageloaded{tikzinput}{
6318 \RequirePackage{stex-tikzinput}
6319 }{}
6320
6321 \bool_if:NT \c_stex_persist_mode_bool {
6322 \input{\jobname.sms}
6323 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6324 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6325 \l_tmpa_str
6326 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6327 \c_stex_mathhub_main_manifest_prop
6328 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6329 }
6330 }
6331 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6332 <*package>
6333 <@@=stex_modules>
6334
6335 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6336
6337 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6338 \begingroup
6339 \stex_module_setup:nn{
6340   ns=\c_stex_metatheory_ns_str,
6341   meta=NONE
6342 }{Metatheory}
6343 \stex_reactivate_macro:N \symdecl
6344 \stex_reactivate_macro:N \notation
6345 \stex_reactivate_macro:N \symdef
6346 \ExplSyntaxOff
6347 \csname stex_suppress_html:n\endcsname{
6348   % is-a (a:A, a \in A, a is an A, etc.)
6349   \symdecl{isa}[args=ai]
6350   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6351   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6352   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6353
6354   % bind (\forall, \Pi, \lambda etc.)
6355   \symdecl{bind}[args=Bi]
6356   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6357   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6358   \notation{bind}[depfun]{\comp( #1 \comp{} \;\to\; ) #2}{##1 \comp, ##2}
6359
6360   % implicit bind
6361   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6362
6363   % dummy variable
6364   \symdecl{dummyvar}
6365   \notation{dummyvar}[underscore]{\comp\_}
6366   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6367 \notation{dummyvar}[dash]{\comp{\rm --}}
6368
6369 %fromto (function space, Hom-set, implication etc.)
6370 \symdecl{fromto}[args=ai]
6371 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6372 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6373
6374 % mapto (lambda etc.)
6375 \symdecl{mapto}[args=Bi]
6376 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6377 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6378 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6379
6380 % function/operator application
6381 \symdecl{apply}[args=ia]
6382 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6383 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6384
6385 % collection of propositions/booleans/truth values
6386 \symdecl{prop}[name=proposition]
6387 \notation{prop}[prop]{\comp{\rm prop}}
6388 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6389
6390 \symdecl{judgmentholds}[args=1]
6391 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6392
6393 % sequences
6394 \symdecl{seqtype}[args=1]
6395 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6396
6397 \symdecl{seqexpr}[args=a]
6398 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6399
6400 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6401 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6402 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6403 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6404 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6405 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6406 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6407 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6408 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6409
6410 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6411 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6412
6413 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6414 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6415 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6416
6417 % letin ("let", local definitions, variable substitution)
6418 \symdecl{letin}[args=bii]
6419 \notation{letin}[let]{\comp{\rm let}\; ;#1\comp{=}#2\; ;\comp{\rm in}\; ;#3}
6420 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6421 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6422
6423 % structures
6424 \symdecl*{module-type}[args=1]
6425 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6426 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6427 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6428
6429 % objects
6430 \symdecl{object}
6431 \notation{object}{\comp{\mathtt{OBJECT}}}
6432
6433 }
6434
6435 % The following are abbreviations in the sTeX corpus that are left over from earlier
6436 % developments. They will eventually be phased out.
6437
6438 \ExplSyntaxOn
6439 \stex_add_to_current_module:n{
6440   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6441   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6442   \def\livar{\csname sequence-index\endcsname[li]}
6443   \def\uivar{\csname sequence-index\endcsname[ui]}
6444   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6445   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6446 }
6447 \__stex_modules_end_module:
6448 \endgroup
6449 \</package>

```

Chapter 36

Tikzinput Implementation

```
6450 <@@=tikzinput>
6451 <*package>
6452
6453 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6454
6455 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6456 \RequirePackage{l3keys2e}
6457
6458 \keys_define:nn { tikzinput } {
6459   image .bool_set:N = \c_tikzinput_image_bool,
6460   image .default:n = false ,
6461   unknown .code:n = {}
6462 }
6463
6464 \ProcessKeysOptions { tikzinput }
6465
6466 \bool_if:NTF \c_tikzinput_image_bool {
6467   \RequirePackage{graphicx}
6468
6469   \providecommand\usetikzlibrary[]{}
6470   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6471 }{
6472   \RequirePackage{tikz}
6473   \RequirePackage{standalone}
6474
6475   \newcommand \tikzinput [2] [] {
6476     \setkeys{Gin}{#1}
6477     \ifx \Gin@ewidth \Gin@exclamation
6478       \ifx \Gin@eheight \Gin@exclamation
6479         \input { #2 }
6480       \else
6481         \resizebox{!}{ \Gin@eheight }{
6482           \input { #2 }
6483         }
6484       \fi
6485     \else
6486       \ifx \Gin@eheight \Gin@exclamation
6487         \resizebox{ \Gin@ewidth }{!}{
```



```

6488         \input { #2 }
6489     }
6490     \else
6491         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6492             \input { #2 }
6493         }
6494     \fi
6495 \fi
6496 }
6497 }
6498
6499 \newcommand \ctikzinput [2] [] {
6500     \begin{center}
6501         \tikzinput [#1] {#2}
6502     \end{center}
6503 }
6504
6505 \@ifpackageloaded{stex}{
6506     \RequirePackage{stex-tikzinput}
6507 }{}
6508
6509 </package>
6510 <*stex>
6511 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6512 \RequirePackage{stex}
6513 \RequirePackage{tikzinput}
6514
6515 \newcommand\mhtikzinput[2] [] {%
6516     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6517     \stex_in_repository:nn\Gin@mhrepos{
6518         \tikzinput[#1]{\mhp@path{##1}{#2}}
6519     }
6520 }
6521 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6522
6523 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6524     \pgfkeys@spdef\pgf@temp{#1}
6525     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6526     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6527     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6528     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6529     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6530     \catcode'\@=11
6531     \catcode'\|=12
6532     \catcode'\$=3
6533     \pgfutil@InputIfFileExists{#2}{-}{-}
6534     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6535     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6536     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6537 }
6538
6539
6540 \newcommand\libusetikzlibrary[1]{

```

```

6541 \prop_if_exist:NF \l_stex_current_repository_prop {
6542   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6543 }
6544 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6545   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6546 }
6547 \seq_clear:N \l__tikzinput_libinput_files_seq
6548 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6549 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6550
6551 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6552   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6553   \IfFileExists{ \l_tmpa_str }{
6554     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6555   }{}
6556   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6557   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6558 }
6559
6560 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6561 \IfFileExists{ \l_tmpa_str }{
6562   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6563 }{}
6564
6565 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6566   \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6567 }{
6568   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6569     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6570       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6571     }
6572   }{
6573     \msg_error:nxxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6574   }
6575 }
6576 }
6577 </stex>

```

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

Chapter 37

document-structure.sty Implementation

```
6578 <*package>
6579 <@@=document_structure>
6580 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6581 \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).

```
6582
6583 \keys_define:nn{ document-structure }{
6584   class      .str_set_x:N = \c_document_structure_class_str,
6585   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6586   unknown    .code:n      = {
6587     \PassOptionsToClass{\CurrentOption}{stex}
6588     \PassOptionsToClass{\CurrentOption}{tikzinput}
6589   }
6590   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6591 }
6592 \ProcessKeysOptions{ document-structure }
6593 \str_if_empty:NT \c_document_structure_class_str {
6594   \str_set:Nn \c_document_structure_class_str {article}
6595 }
6596 \str_if_empty:NT \c_document_structure_topsect_str {
6597   \str_set:Nn \c_document_structure_topsect_str {section}
6598 }
```

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

```
6599 \RequirePackage{xspace}
6600 \RequirePackage{comment}
6601 \RequirePackage{stex}
6602 \AddToHook{begindocument}{}
```

```

6603 \ltx@ifpackageloaded{babel}{
6604     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6605     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6606         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6607     }
6608 }{}
6609 }

```

`\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.

```

6610 \int_new:N \l_document_structure_section_level_int
6611 \str_case:VnF \c_document_structure_topsect_str {
6612     {part}}{
6613         \int_set:Nn \l_document_structure_section_level_int {0}
6614     }
6615     {chapter}{
6616         \int_set:Nn \l_document_structure_section_level_int {1}
6617     }
6618 }{
6619     \str_case:VnF \c_document_structure_class_str {
6620         {book}{
6621             \int_set:Nn \l_document_structure_section_level_int {0}
6622         }
6623         {report}{
6624             \int_set:Nn \l_document_structure_section_level_int {0}
6625         }
6626     }{
6627         \int_set:Nn \l_document_structure_section_level_int {2}
6628     }
6629 }

```

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

```

6630 \def\current@section@level{document}%
6631 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6632 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6633 \cs_new_protected:Npn \skipfragment {

```

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

```

6634 \ifcase\l_document_structure_section_level_int
6635 \or\stepcounter{part}
6636 \or\stepcounter{chapter}
6637 \or\stepcounter{section}
6638 \or\stepcounter{subsection}
6639 \or\stepcounter{subsubsection}
6640 \or\stepcounter{paragraph}
6641 \or\stepcounter{subparagraph}
6642 \fi
6643 }

```

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

blindfragment

```

6644 \newcommand\at@begin@blindsfragment[1]{
6645 \newenvironment{blindfragment}
6646 {
6647 \int_incr:N\l_document_structure_section_level_int
6648 \at@begin@blindsfragment\l_document_structure_section_level_int
6649 }{}

```

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

```

6650 \newcommand\sfragment@nonum[2]{
6651 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6652 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6653 }

```

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

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

```

6654 \newcommand\sfragment@num[2]{
6655 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6656 \@nameuse{#1}{#2}
6657 }{
6658 \cs_if_exist:NTF\rdfmata@sectioning{
6659 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6660 }{
6661 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6662 }
6663 }
6664 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6665 }

```

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

sfragment

```

6666 \keys_define:nn { document-structure / sfragment }{
6667 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6668 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6669 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6670 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6671 srccite        .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6672 type          .tl_set:N     = \l__document_structure_sfragment_type_tl,
6673 short         .tl_set:N     = \l__document_structure_sfragment_short_tl,
6674 display       .tl_set:N     = \l__document_structure_sfragment_display_tl,
6675 intro         .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6676 imports       .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6677 loadmodules   .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6678 }
6679 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6680   \str_clear:N \l__document_structure_sfragment_id_str
6681   \str_clear:N \l__document_structure_sfragment_date_str
6682   \clist_clear:N \l__document_structure_sfragment_creators_clist
6683   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6684   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6685   \tl_clear:N \l__document_structure_sfragment_type_tl
6686   \tl_clear:N \l__document_structure_sfragment_short_tl
6687   \tl_clear:N \l__document_structure_sfragment_display_tl
6688   \tl_clear:N \l__document_structure_sfragment_imports_tl
6689   \tl_clear:N \l__document_structure_sfragment_intro_tl
6690   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6691   \keys_set:nn { document-structure / sfragment } { #1 }
6692 }

```

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.

```

6693 \newif@if@mainmatter\@mainmattertrue
6694 \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.

```

6695 \keys_define:nn { document-structure / sectioning }{
6696   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
6697   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
6698   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6699   clear     .default:n   = {true}                                ,
6700   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
6701   num       .default:n   = {true}
6702 }
6703 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6704   \str_clear:N \l__document_structure_sect_name_str
6705   \str_clear:N \l__document_structure_sect_ref_str
6706   \bool_set_false:N \l__document_structure_sect_clear_bool
6707   \bool_set_false:N \l__document_structure_sect_num_bool
6708   \keys_set:nn { document-structure / sectioning } { #1 }
6709 }
6710 \newcommand\omdoc@sectioning[3][]{ }
6711   \l__document_structure_sect_args:n {#1 }
6712   \let\omdoc@sect@name\l__document_structure_sect_name_str
6713   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6714   \if@mainmatter% numbering not overridden by frontmatter, etc.
6715     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6716     \sfragment@num{#2}{#3}
6717   }{
6718     \sfragment@nonum{#2}{#3}
6719   }
6720   \def\current@section@level{\omdoc@sect@name}
6721 \else
6722   \sfragment@nonum{#2}{#3}
6723 \fi
6724 }% 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.

```

6725 \newcommand\sfragment@redefine@addtocontents[1]{%
6726   %\edef\__document_structureimport{#1}%
6727   %\@for\@I:=\__document_structureimport\do{%
6728     %\edef\@path{\csname module@\@I @path\endcsname}%
6729     %\@ifundefined{tf@toc}\relax%
6730     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6731   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6732   %\def\addcontentsline##1##2##3{%
6733     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6734   %\else% hyperref.sty not loaded
6735   %\def\addcontentsline##1##2##3{%
6736     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6737   %\fi
6738   }% hyperref.sty loaded?

```

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

```

6739 \newenvironment{sfragment}[2] []% keys, title
6740 {
6741   \__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.

```

6742   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6743
6744   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6745     \sfragment@redefine@addtocontents{
6746       %\@ifundefined{module@id}\used@modules%
6747       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6748     }
6749   }

```

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

```

6750
6751   \stex_document_title:n { #2 }
6752
6753   \int_incr:N\l__document_structure_section_level_int
6754   \ifcase\l__document_structure_section_level_int
6755     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6756     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6757 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6758 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6759 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6760 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6761 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6762 \fi
6763 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6764 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6765   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6766 }
6767 }% for customization
6768 {}

```

and finally, we localize the sections

```

6769 \newcommand\omdoc@part@kw{Part}
6770 \newcommand\omdoc@chapter@kw{Chapter}
6771 \newcommand\omdoc@section@kw{Section}
6772 \newcommand\omdoc@subsection@kw{Subsection}
6773 \newcommand\omdoc@subsubsection@kw{Subsubsubsection}
6774 \newcommand\omdoc@paragraph@kw{paragraph}
6775 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 Front and Backmatter

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

`\printindex`

```

6776 \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).

```

6777 \cs_if_exist:NTF\frontmatter{
6778   \let\__document_structure_orig_frontmatter\frontmatter
6779   \let\frontmatter\relax
6780 }{
6781   \tl_set:Nn\__document_structure_orig_frontmatter{
6782     \clearpage
6783     \@mainmatterfalse
6784     \pagenumbering{roman}
6785   }
6786 }
6787 \cs_if_exist:NTF\backmatter{
6788   \let\__document_structure_orig_backmatter\backmatter
6789   \let\backmatter\relax
6790 }{
6791   \tl_set:Nn\__document_structure_orig_backmatter{
6792     \clearpage
6793     \@mainmatterfalse

```



```

6794     \pagenumbering{roman}
6795   }
6796 }

```

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.

```

6797 \newenvironment{frontmatter}{
6798   \_document_structure_orig_frontmatter
6799 }{
6800   \cs_if_exist:NTF\mainmatter{
6801     \mainmatter
6802   }{
6803     \clearpage
6804     \@mainmattertrue
6805     \pagenumbering{arabic}
6806   }
6807 }

```

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

```

6808 \newenvironment{backmatter}{
6809   \_document_structure_orig_backmatter
6810 }{
6811   \cs_if_exist:NTF\mainmatter{
6812     \mainmatter
6813   }{
6814     \clearpage
6815     \@mainmattertrue
6816     \pagenumbering{arabic}
6817   }
6818 }

```

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

```

6819 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6820 \def \c__document_structure_document_str{document}
6821 \newcommand\afterprematurestop{}
6822 \def\prematurestop@endsfragment{
6823   \unless\ifx\@currenvir\c__document_structure_document_str
6824     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter{\expandafter
6825     \expandafter\prematurestop@endsfragment
6826   \fi
6827 }
6828 \providecommand\prematurestop{
6829   \message{Stopping~sTeX~processing~prematurely}
6830   \prematurestop@endsfragment
6831   \afterprematurestop
6832   \end{document}
6833 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
6834 \RequirePackage{etoolbox}
6835 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6836 \newrobustcmd\useSGvar[1]{%
6837   \@ifundefined{sTeX@Gvar@#1}
6838   {\PackageError{document-structure}
6839     {The sTeX Global variable #1 is undefined}
6840     {set it with \protect\setSGvar}}
6841   \@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.

```
6842 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6843   \@ifundefined{sTeX@Gvar@#1}
6844   {\PackageError{document-structure}
6845     {The sTeX Global variable #1 is undefined}
6846     {set it with \protect\setSGvar}}
6847   {\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.

```
6848 \*cls)
6849 \@@=notesslides)
6850 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6851 \RequirePackage{13keys2e}
6852
6853 \keys_define:nn{notesslides / cls}{
6854   class .str_set_x:N = \c__notesslides_class_str,
6855   notes .bool_set:N = \c__notesslides_notes_bool ,
6856   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6857   docopt .str_set_x:N = \c__notesslides_docopt_str,
6858   unknown .code:n = {
6859     \PassOptionsToPackage{\CurrentOption}{document-structure}
6860     \PassOptionsToClass{\CurrentOption}{beamer}
6861     \PassOptionsToPackage{\CurrentOption}{notesslides}
6862     \PassOptionsToPackage{\CurrentOption}{stex}
6863   }
6864 }
6865 \ProcessKeysOptions{ notesslides / cls }
6866
6867 \str_if_empty:NF \c__notesslides_class_str {
6868   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6869 }
6870
6871 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6872   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6873 }
6874 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6875   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6876 }
6877
6878 \RequirePackage{stex}
```

```

6879 \stex_html_backend:T {
6880   \bool_set_true:N\c__notesslides_notes_bool
6881 }
6882
6883 \bool_if:NTF \c__notesslides_notes_bool {
6884   \PassOptionsToPackage{notes=true}{notesslides}
6885 }{
6886   \PassOptionsToPackage{notes=false}{notesslides}
6887 }
6888 \</cls>

```

now we do the same for the notesslides package.

```

6889 <*package>
6890 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6891 \RequirePackage{13keys2e}
6892
6893 \keys_define:nn{notesslides / pkg}{
6894   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6895   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6896   notes            .bool_set:N = \c__notesslides_notes_bool ,
6897   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6898   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6899   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6900   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6901   nopproblems      .bool_set:N = \c__notesslides_nopproblems_bool,
6902   unknown          .code:n      = {
6903     \PassOptionsToClass{\CurrentOption}{stex}
6904     \PassOptionsToClass{\CurrentOption}{tikzinput}
6905   }
6906 }
6907 \ProcessKeysOptions{ notesslides / pkg }
6908
6909 \RequirePackage{stex}
6910 \stex_html_backend:T {
6911   \bool_set_true:N\c__notesslides_notes_bool
6912 }
6913
6914 \newif\ifnotes
6915 \bool_if:NTF \c__notesslides_notes_bool {
6916   \notesttrue
6917 }{
6918   \notesfalse
6919 }
6920

```

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

```

6921 \str_if_empty:NTF \c__notesslides_topsect_str {
6922   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6923 }{
6924   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6925 }
6926 \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6927 \</package>

```

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

```

6928 <*cls>
6929 \bool_if:NTF \c__notesslides_notes_bool {
6930   \str_if_empty:NT \c__notesslides_class_str {
6931     \str_set:Nn \c__notesslides_class_str {article}
6932   }
6933   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6934     {\c__notesslides_class_str}
6935 }{
6936   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6937   \newcounter{Item}
6938   \newcounter{paragraph}
6939   \newcounter{subparagraph}
6940   \newcounter{Hfootnote}
6941 }
6942 \RequirePackage{document-structure}

```

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

```

6943 \RequirePackage{notesslides}
6944 </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.

```

6945 <*package>
6946 \bool_if:NT \c__notesslides_notes_bool {
6947   \RequirePackage{a4wide}
6948   \RequirePackage{marginnote}
6949   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6950   \RequirePackage{mdframed}
6951   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6952   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6953 }
6954 \RequirePackage{stex-tikzinput}
6955 \RequirePackage{etoolbox}
6956 \RequirePackage{amssymb}
6957 \RequirePackage{amsmath}
6958 \RequirePackage{comment}
6959 \RequirePackage{textcomp}
6960 \RequirePackage{url}
6961 \RequirePackage{graphicx}
6962 \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

```

6963 \bool_if:NT \c__notesslides_notes_bool {
6964   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6965 }
6966
6967
6968 \NewDocumentCommand \libusetheme {0{} m} {
6969   \bool_if:NTF \c__notesslides_notes_bool {
6970     \libusepackage[#1]{beamernotestheme#2}
6971   }{
6972     \libusepackage[#1]{beamertheme#2}
6973   }
6974 }

```

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

```

6975 \newcounter{slide}
6976 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6977 \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.

```

6978 \bool_if:NTF \c__notesslides_notes_bool {
6979   \renewenvironment{note}{\ignorespaces}{}
6980 }{
6981   \excludcomment{note}
6982 }

```

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.

```

6983 \bool_if:NT \c__notesslides_notes_bool {
6984   \newlength{\slideframewidth}
6985   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6986 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6987   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6988     \bool_set_true:N #1
6989   }{
6990     \bool_set_false:N #1
6991   }
6992 }
6993 \keys_define:nn{notesslides / frame}{
6994   label .str_set_x:N = \l__notesslides_frame_label_str,
6995   allowframebreaks .code:n = {
6996     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6997   },
6998   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.

```

6999     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
7000   },
7001   fragile           .code:n      = {
7002     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
7003   },
7004   shrink           .code:n      = {
7005     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
7006   },
7007   squeeze          .code:n      = {
7008     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
7009   },
7010   t                 .code:n      = {
7011     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
7012   },
7013   unknown          .code:n      = {}
7014 }
7015 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
7016   \str\_clear:N \l\_notesslides\_frame\_label\_str
7017   \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
7018   \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
7019   \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
7020   \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
7021   \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
7022   \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
7023   \keys\_set:nn { notesslides / frame }{ #1 }
7024 }

```

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

```

7025 \renewenvironment{frame}[1][]{
7026   \_notesslides\_frame\_args:n{#1}
7027   \sffamily
7028   \stepcounter{slide}
7029   \def\@currentlabel{\theslide}
7030   \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
7031     \label{\l\_notesslides\_frame\_label\_str}
7032   }

```

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

```

7033 \def\itemize@level{outer}
7034 \def\itemize@outer{outer}
7035 \def\itemize@inner{inner}
7036 \renewcommand\newpage{\addtocounter{framenumber}{1}}
7037 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7038 \renewenvironment{itemize}{
7039   \ifx\itemize@level\itemize@outer
7040     \def\itemize@label{$\rhd$}
7041   \fi
7042   \ifx\itemize@level\itemize@inner
7043     \def\itemize@label{$\scriptstyle\rhd$}
7044   \fi
7045   \begin{list}
7046     {\itemize@label}
7047     {\setlength{\labelsep}{.3em}
7048      \setlength{\labelwidth}{.5em}
7049      \setlength{\leftmargin}{1.5em}

```

```

7050     }
7051     \edef\itemize@level{\itemize@inner}
7052   }{
7053     \end{list}
7054   }

```

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

```

7055     \stex_html_backend:TF {
7056       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7057         \mdf@patchamsthm
7058       }{
7059         \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7060       }
7061     }{
7062       \stex_html_backend:TF {
7063         \miko@slidelabel\egroup\end{stex_annotate_env}
7064       }{\medskip\miko@slidelabel\end{mdframed}}
7065     }

```

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

`\frametitle`

```

7066     \renewcommand{\frametitle}[1]{
7067       \stex_document_title:n { #1 }
7068       {\Large\bf\sf\color{blue}{#1}}\medskip
7069     }
7070   }

```

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

EdN:19

`\pause` 19

```

7071     \bool_if:NT \c__notesslides_notes_bool {
7072       \newcommand\pause{}
7073     }

```

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

`nparagraph`

```

7074     \bool_if:NTF \c__notesslides_notes_bool {
7075       \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7076     }{
7077       \excludecomment{nparagraph}
7078     }

```

`nfragment`

```

7079     \bool_if:NTF \c__notesslides_notes_bool {
7080       \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7081     }{
7082       \excludecomment{nfragment}
7083     }

```

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

ndefinition

```
7084 \bool_if:NTF \c__notesslides_notes_bool {
7085   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7086 }{
7087   \excludecomment{ndefinition}
7088 }
```

nassertion

```
7089 \bool_if:NTF \c__notesslides_notes_bool {
7090   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7091 }{
7092   \excludecomment{nassertion}
7093 }
```

nsproof

```
7094 \bool_if:NTF \c__notesslides_notes_bool {
7095   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
7096 }{
7097   \excludecomment{nproof}
7098 }
```

nexample

```
7099 \bool_if:NTF \c__notesslides_notes_bool {
7100   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
7101 }{
7102   \excludecomment{nexample}
7103 }
```

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

```
7104 \def\inputref@preskip{\smallskip}
7105 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7106 \let\orig@inputref\inputref
7107 \def\inputref{\@ifstar\ninputref\orig@inputref}
7108 \newcommand\ninputref[2] [] {
7109   \bool_if:NT \c__notesslides_notes_bool {
7110     \orig@inputref[#1]{#2}
7111   }
7112 }
```

(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.

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

```

7113 \newlength{\slideologoheight}
7114
7115 \bool_if:NTF \c_notesslides_notes_bool {
7116   \setlength{\slideologoheight}{.4cm}
7117 }{
7118   \setlength{\slideologoheight}{1cm}
7119 }
7120 \newsavebox{\slideologo}
7121 \sbox{\slideologo}{\text{\LaTeX}}
7122 \newrobustcmd{\setslideologo}[1]{
7123   \sbox{\slideologo}{\includegraphics[height=\slideologoheight]{#1}}
7124 }

```

(End definition for `\setslideologo`. 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.

```

7125 \def\source{Michael Kohlhase}% customize locally
7126 \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.

```

7127 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7128 \newsavebox{\cclogo}
7129 \sbox{\cclogo}{\includegraphics[height=\slideologoheight]{stex-cc_somerights}}
7130 \newif\ifcchref\cchreffalse
7131 \AtBeginDocument{
7132   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7133 }
7134 \def\licensing{
7135   \ifcchref
7136     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7137   \else
7138     {\usebox{\cclogo}}
7139   \fi
7140 }
7141 \newrobustcmd{\setlicensing}[2][]{
7142   \def\@url{#1}
7143   \sbox{\cclogo}{\includegraphics[height=\slideologoheight]{#2}}
7144   \ifx\@url\@empty
7145     \def\licensing{{\usebox{\cclogo}}}
7146   \else
7147     \def\licensing{
7148       \ifcchref
7149         \href{#1}{\usebox{\cclogo}}
7150       \else
7151         {\usebox{\cclogo}}
7152       \fi

```

```

7153     }
7154     \fi
7155 }

```

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

EdN:20

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

```

7156 \newrobustcmd\miko@slidelabel{
7157   \vbox to \slidelogoheight{
7158     \vss\hbox to \slidewidth
7159     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7160   }
7161 }

```

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

38.4 Frame Images

`\frameimage` We have to make sure that the width is overwritten, for that we check the `\Gin@ewidth` macro from the `graphicx` package. We also add the `label` key.

```

7162 \def\Gin@mhrepos{}
7163 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7164 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7165 \newrobustcmd\frameimage[2][]{
7166   \stepcounter{slide}
7167   \bool_if:NT \c__notesslides_frameimages_bool {
7168     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7169     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7170     \begin{center}
7171       \bool_if:NTF \c__notesslides_fiboxed_bool {
7172         \fbox{
7173           \ifx\Gin@ewidth\@empty
7174             \ifx\Gin@mhrepos\@empty
7175               \mhgraphics[width=\slidewidth,#1]{#2}
7176             \else
7177               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7178             \fi
7179           \else% Gin@ewidth empty
7180             \ifx\Gin@mhrepos\@empty
7181               \mhgraphics[#1]{#2}
7182             \else
7183               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7184             \fi
7185           \fi% Gin@ewidth empty
7186         }
7187       }{
7188         \ifx\Gin@ewidth\@empty
7189           \ifx\Gin@mhrepos\@empty
7190             \mhgraphics[width=\slidewidth,#1]{#2}
7191           \else
7192             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7193           \fi

```

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

```

7194         \ifx\Gin@mhrepos\@empty
7195             \mhgraphics[#1]{#2}
7196         \else
7197             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7198         \fi
7199     \fi% Gin@ewidth empty
7200 }
7201 \end{center}
7202 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7203 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7204 }
7205 } % 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.

```

7206 \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

```

7207 \AddToHook{begindocument}{
7208     \definecolor{green}{rgb}{0,.5,0}
7209     \definecolor{purple}{cmyk}{.3,1,0,.17}
7210 }

```

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`.

```

7211 % \def\STpresent#1{\textcolor{blue}{#1}}
7212 \def\defemph#1{\textcolor{magenta}{#1}}
7213 \def\symrefemph#1{\textcolor{cyan}{#1}}
7214 \def\compemph#1{\textcolor{blue}{#1}}
7215 \def\titleemph#1{\textcolor{blue}{#1}}
7216 \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.

```

7217 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7218 \def\smalltextwarning{
7219     \pgfuseimage{miko@small@dbend}
7220     \xspace
7221 }
7222 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7223 \newrobustcmd\textwarning{
7224     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7225     \xspace
7226 }
7227 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}

```

```

7228 \newrobustcmd\bigtextwarning{
7229   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7230   \xspace
7231 }
(End definition for \textwarning. This function is documented on page 55.)
7232 \newrobustcmd\putgraphicsat[3]{
7233   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7234 }
7235 \newrobustcmd\putat[2]{
7236   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7237 }

```

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`.

```

7238 \stex_html_backend:F {
7239   \bool_if:NT \c__notesslides_sectocframes_bool {
7240     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7241       \newcounter{chapter}\counterwithin*{section}{chapter}
7242     }{
7243       \str_if_eq:VnT\__notesslidesstopsect{chapter}{
7244         \newcounter{chapter}\counterwithin*{section}{chapter}
7245       }
7246     }
7247   }
7248 }
\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
7249 \def\part@prefix{}
7250 \@ifpackageloaded{document-structure}{}{
7251   \str_case:VnF \__notesslidesstopsect {
7252     {part}{
7253       \int_set:Nn \l_document_structure_section_level_int {0}
7254       \def\thesection{\arabic{chapter}.\arabic{section}}
7255       \def\part@prefix{\arabic{chapter}.}
7256     }
7257     {chapter}{
7258       \int_set:Nn \l_document_structure_section_level_int {1}
7259       \def\thesection{\arabic{chapter}.\arabic{section}}
7260       \def\part@prefix{\arabic{chapter}.}
7261     }
7262   }{
7263     \int_set:Nn \l_document_structure_section_level_int {2}
7264     \def\part@prefix{}
7265   }
7266 }
7267
7268 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `sfragment` environment that choses the L^AT_EX sectioning macros according to \section@level.

`sfragment`

```

7269 \renewenvironment{sfragment}[2][]{
7270   \_document_structure_sfragment_args:n { #1 }
7271   \int_incr:N \l_document_structure_section_level_int
7272   \bool_if:NT \c_notesslides_sectocframes_bool {
7273     \stepcounter{slide}
7274     \begin{frame}[noframenumbering]
7275     \vfill\Large\centering
7276     \red{
7277       \ifcase\l_document_structure_section_level_int\or
7278         \stepcounter{part}
7279         \def\_notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7280         \def\currentsectionlevel{\omdoc@part@kw}
7281       \or
7282         \stepcounter{chapter}
7283         \def\_notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7284         \def\currentsectionlevel{\omdoc@chapter@kw}
7285       \or
7286         \stepcounter{section}
7287         \def\_notesslideslabel{\part@prefix\arabic{section}}
7288         \def\currentsectionlevel{\omdoc@section@kw}
7289       \or
7290         \stepcounter{subsection}
7291         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7292         \def\currentsectionlevel{\omdoc@subsection@kw}
7293       \or
7294         \stepcounter{subsubsection}
7295         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7296         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7297       \or
7298         \stepcounter{paragraph}
7299         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7300         \def\currentsectionlevel{\omdoc@paragraph@kw}
7301       \else
7302         \def\_notesslideslabel{}
7303         \def\currentsectionlevel{\omdoc@paragraph@kw}
7304       \fi% end ifcase
7305       \_notesslideslabel%\sref@label@id\_notesslideslabel
7306       \quad #2%
7307     }%
7308     \vfill%
7309     \end{frame}%
7310   }
7311   \str_if_empty:NF \l_document_structure_sfragment_id_str {
7312     \stex_ref_new_doc_target:n\l_document_structure_sfragment_id_str
7313   }
7314 }{}
7315 }
```

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

```

7316 \def\inserttheorembodyfont{\normalfont}
7317 %\bool_if:NF \c__notesslides_notes_bool {
7318 % \defbeamertemplate{theorem begin}{miko}
7319 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7320 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7321 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7322 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

7323 % \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.

```

7324 % \expandafter\def\csname Parent2\endcsname{}
7325 %}
7326
7327 \AddToHook{begindocument}{ % this does not work for some reason
7328 \setbeamertemplate{theorems}[ams style]
7329 }
7330 \bool_if:NT \c__notesslides_notes_bool {
7331 \renewenvironment{columns}[1][{}%
7332 \par\noindent%
7333 \begin{minipage}%
7334 \slidewidth\centering\leavevmode%
7335 }{}%
7336 \end{minipage}\par\noindent%
7337 }%
7338 \newsavebox\columnbox%
7339 \renewenvironment<>{column}[2][{}%
7340 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7341 }{}%
7342 \end{minipage}\end{lrbox}\usebox\columnbox%
7343 }%
7344 }
7345 \bool_if:NTF \c__notesslides_noproblems_bool {
7346 \newenvironment{problems}{}{}
7347 }{
7348 \excludecomment{problems}
7349 }
```

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.

```

7350 \gdef\printexcursions{}
7351 \newcommand\excursionref[2]{% label, text
7352 \bool_if:NT \c__notesslides_notes_bool {
7353 \begin{sparagraph}[title=Excursion]
7354 #2 \sref[fallback=the appendix]{#1}.
7355 \end{sparagraph}}
```

```

7356 }
7357 }
7358 \newcommand\activate@excursion[2][]{
7359   \gappto\printexcursions{\inputref{#1}{#2}}
7360 }
7361 \newcommand\excursion[4][]{% repos, label, path, text
7362   \bool_if:NT \c__notesslides_notes_bool {
7363     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7364   }
7365 }

```

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

\excursiongroup

```

7366 \keys_define:nn{notesslides / excursiongroup }{
7367   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7368   intro       .tl_set:N    = \l__notesslides_excursion_intro_tl,
7369   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7370 }
7371 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7372   \tl_clear:N \l__notesslides_excursion_intro_tl
7373   \str_clear:N \l__notesslides_excursion_id_str
7374   \str_clear:N \l__notesslides_excursion_mhrepos_str
7375   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7376 }
7377 \newcommand\excursiongroup[1][]{
7378   \__notesslides_excursion_args:n{ #1 }
7379   \ifdefempty\printexcursions{}% only if there are excursions
7380   {\begin{note}
7381     \begin{sfragment}[#1]{Excursions}%
7382     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7383       \inputref[\l__notesslides_excursion_mhrepos_str]{
7384         \l__notesslides_excursion_intro_tl
7385       }
7386     }
7387     \printexcursions%
7388     \end{sfragment}
7389   }\end{note}}
7390 }
7391 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7392 \</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.

```
7393 <*package>
7394 <@@=problems>
7395 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7396 \RequirePackage{l3keys2e,stex}
7397
7398 \keys_define:nn { problem / pkg }{
7399   notes      .default:n    = { true },
7400   notes      .bool_set:N   = \c__problems_notes_bool,
7401   gnotes     .default:n    = { true },
7402   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7403   hints      .default:n    = { true },
7404   hints      .bool_set:N   = \c__problems_hints_bool,
7405   solutions  .default:n    = { true },
7406   solutions  .bool_set:N   = \c__problems_solutions_bool,
7407   pts        .default:n    = { true },
7408   pts        .bool_set:N   = \c__problems_pts_bool,
7409   min        .default:n    = { true },
7410   min        .bool_set:N   = \c__problems_min_bool,
7411   boxed      .default:n    = { true },
7412   boxed      .bool_set:N   = \c__problems_boxed_bool,
7413   unknown    .code:n       = {}
7414 }
7415 \newif\ifsolutions
7416
7417 \ProcessKeysOptions{ problem / pkg }
7418 \bool_if:NTF \c__problems_solutions_bool {
7419   \solutionstrue
7420 }{
7421   \solutionsfalse
7422 }
```

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

```
7423 \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.

```
7424 \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.

```
7425 \def\prob@problem@kw{Problem}
7426 \def\prob@solution@kw{Solution}
7427 \def\prob@hint@kw{Hint}
7428 \def\prob@note@kw{Note}
7429 \def\prob@gnote@kw{Grading}
7430 \def\prob@pt@kw{pt}
7431 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7432 \AddToHook{begindocument}{
7433   \ltx@ifpackageloaded{babel}{
7434     \makeatletter
7435     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7436     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7437       \input{problem-ngerman.ldf}
7438     }
7439     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7440       \input{problem-finnish.ldf}
7441     }
7442     \clist_if_in:NnT \l_tmpa_clist {french}{
7443       \input{problem-french.ldf}
7444     }
7445     \clist_if_in:NnT \l_tmpa_clist {russian}{
7446       \input{problem-russian.ldf}
7447     }
7448     \makeatother
7449   }{}
7450 }
```

39.2 Problems and Solutions

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

```
7451 \keys_define:nn{ problem / problem }{
7452   id      .str_set_x:N = \l__problems_prob_id_str,
7453   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7454   min     .tl_set:N    = \l__problems_prob_min_tl,
7455   title   .tl_set:N    = \l__problems_prob_title_tl,
7456   type    .tl_set:N    = \l__problems_prob_type_tl,
7457   imports .tl_set:N    = \l__problems_prob_imports_tl,
7458   name    .str_set_x:N = \l__problems_prob_name_str,
7459   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7460 }
7461 \cs_new_protected:Nn \__problems_prob_args:n {
7462   \str_clear:N \l__problems_prob_id_str
7463   \str_clear:N \l__problems_prob_name_str
7464   \tl_clear:N \l__problems_prob_pts_tl
7465   \tl_clear:N \l__problems_prob_min_tl
7466   \tl_clear:N \l__problems_prob_title_tl
7467   \tl_clear:N \l__problems_prob_type_tl
7468   \tl_clear:N \l__problems_prob_imports_tl
7469   \int_zero_new:N \l__problems_prob_refnum_int
7470   \keys_set:nn { problem / problem }{ #1 }
7471   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7472     \let\l__problems_prob_refnum_int\undefined
7473   }
7474 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7475 \newcounter{problem}[section]
7476 \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.

```

7477 \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

```

7478 \newcommand\prob@number{
7479   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7480     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7481   }{
7482     \int_if_exist:NTF \l__problems_prob_refnum_int {
7483       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7484     }{
7485       \prob@label\theproblem
7486     }
7487   }
7488 }

```

(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.

```

7489 \newcommand\prob@title[3]{%
7490   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7491     #2 \l__problems_inclprob_title_tl #3
7492   }{
7493     \tl_if_exist:NTF \l__problems_prob_title_tl {
7494       #2 \l__problems_prob_title_tl #3
7495     }{
7496       #1

```

```

7497     }
7498   }
7499 }

```

(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.

```

7500 \def\prob@heading{
7501   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7502   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7503 }

```

(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`

```

7504 \newenvironment{sproblem}[1][{}]{
7505   \__problems_prob_args:n{#1}%\sref@target%
7506   \@in@omtexttrue% we are in a statement (for inline definitions)
7507   \stepcounter{problem}\record@problem
7508   \def\current@section@level{\prob@problem@kw}
7509
7510   \str_if_empty:NT \l__problems_prob_name_str {
7511     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7512     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7513     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7514   }
7515
7516   \stex_if_do_html:T{
7517     \tl_if_empty:NF \l__problems_prob_title_tl {
7518       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7519     }
7520   }
7521
7522   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7523
7524   \stex_reactivate_macro:N \STEXexport
7525   \stex_reactivate_macro:N \importmodule
7526   \stex_reactivate_macro:N \symdecl
7527   \stex_reactivate_macro:N \notation
7528   \stex_reactivate_macro:N \symdef
7529
7530   \stex_if_do_html:T{
7531     \begin{stex_annotate_env} {problem} {
7532       \l_stex_module_ns_str ? \l_stex_module_name_str
7533     }
7534
7535     \stex_annotate_invisible:nnn{header}{} {
7536       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7537     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7538     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7539         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7540     }
7541 }
7542 }
7543
7544 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7545
7546
7547 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7548     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7549 }{
7550     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7551 }
7552 \str_if_exist:NTF \l__problems_inclprob_id_str {
7553     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7554 }{
7555     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7556 }
7557
7558
7559 \stex_if_smsmode:F {
7560     \clist_set:No \l_tmpa_clist \sproblemtype
7561     \tl_clear:N \l_tmpa_tl
7562     \clist_map_inline:Nn \l_tmpa_clist {
7563         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7564             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7565         }
7566     }
7567     \tl_if_empty:NTF \l_tmpa_tl {
7568         \__problems_sproblem_start:
7569     }{
7570         \l_tmpa_tl
7571     }
7572 }
7573 \stex_ref_new_doc_target:n \sproblemid
7574 \stex_smsmode_do:
7575 }{
7576     \__stex_modules_end_module:
7577     \stex_if_smsmode:F{
7578         \clist_set:No \l_tmpa_clist \sproblemtype
7579         \tl_clear:N \l_tmpa_tl
7580         \clist_map_inline:Nn \l_tmpa_clist {
7581             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7582                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7583             }
7584         }
7585         \tl_if_empty:NTF \l_tmpa_tl {
7586             \__problems_sproblem_end:
7587         }{
7588             \l_tmpa_tl
7589         }
7590     }

```

```

7591 \stex_if_do_html:T{
7592   \end{stex_annotate_env}
7593 }
7594
7595 \smallskip
7596 }
7597
7598 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7599
7600
7601
7602 \cs_new_protected:Nn \__problems_sproblem_start: {
7603   \par\noindent\textbf{\prob@heading\show@pts\show@min\\\ignorespacesandpars
7604 }
7605 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7606
7607 \newcommand\stexpatchproblem[3][] {
7608   \str_set:Nx \l_tmpa_str{ #1 }
7609   \str_if_empty:NTF \l_tmpa_str {
7610     \tl_set:Nn \__problems_sproblem_start: { #2 }
7611     \tl_set:Nn \__problems_sproblem_end: { #3 }
7612   }{
7613     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7614     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7615   }
7616 }
7617
7618
7619 \bool_if:NT \c__problems_boxed_bool {
7620   \surroundwithmdframed{problem}
7621 }

```

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

```

7622 \def\record@problem{
7623   \protected@write\@auxout{}
7624   {
7625     \string\@problem{\prob@number}
7626     {
7627       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7628         \l__problems_inclprob_pts_tl
7629       }{
7630         \l__problems_prob_pts_tl
7631       }
7632     }%
7633     {
7634       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7635         \l__problems_inclprob_min_tl
7636       }{
7637         \l__problems_prob_min_tl
7638       }
7639     }
7640   }
7641 }

```

(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).

```
7642 \def\@problem#1#2#3{}
```

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

`solution` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7643 \keys_define:nn { problem / solution }{
7644   id          .str_set_x:N = \l__problems_solution_id_str ,
7645   for         .tl_set:N    = \l__problems_solution_for_tl ,
7646   height      .dim_set:N   = \l__problems_solution_height_dim ,
7647   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7648   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7649   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7650 }
7651 \cs_new_protected:Nn \__problems_solution_args:n {
7652   \str_clear:N \l__problems_solution_id_str
7653   \tl_clear:N \l__problems_solution_for_tl
7654   \tl_clear:N \l__problems_solution_srccite_tl
7655   \clist_clear:N \l__problems_solution_creators_clist
7656   \clist_clear:N \l__problems_solution_contributors_clist
7657   \dim_zero:N \l__problems_solution_height_dim
7658   \keys_set:nn { problem / solution }{ #1 }
7659 }
```

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

```
7660 \newcommand\@startsolution[1][]{
7661   \__problems_solution_args:n { #1 }
7662   \@in@omtexttrue% we are in a statement.
7663   \bool_if:NF \c__problems_boxed_bool { \hrule }
7664   \smallskip\noindent
7665   {\textbf{\prob@solution@kw :}\enspace}
7666   \begin{small}
7667   \def\current@section@level{\prob@solution@kw}
7668   \ignorespacesandpars
7669 }
```

`\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.

```
7670 \box_new:N \l__problems_solution_box
7671 \newenvironment{solution}[1][]{
7672   \stex_html_backend:TF{
7673     \stex_if_do_html:T{
7674       \begin{stex_annotate_env}{solution}{}}
7675     }
7676   }{
7677     \setbox\l__problems_solution_box\vbox\bgroup
7678     \par\smallskip\hrule\smallskip
7679     \noindent\textbf{Solution:}~
7680   }
7681 }{
7682   \stex_html_backend:TF{
```

```

7683 \stex_if_do_html:T{
7684 \end{stex_annotate_env}
7685 }
7686 }{
7687 \smallskip\hrule
7688 \egroup
7689 \bool_if:NT \c__problems_solutions_bool {
7690 \box\l__problems_solution_box
7691 }
7692 }
7693 }
7694
7695 \newcommand\startsolutions{
7696 \bool_set_true:N \c__problems_solutions_bool
7697 % \specialcomment{solution}{\@startsolution}{
7698 % \bool_if:NF \c__problems_boxed_bool {
7699 % \hrule\medskip
7700 % }
7701 % \end{small}%
7702 % }
7703 % \bool_if:NT \c__problems_boxed_bool {
7704 % \surroundwithmdframed{solution}
7705 % }
7706 }

```

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

\stopsolutions

```

7707 \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.

```

7708 \ifsolutions
7709 \startsolutions
7710 \else
7711 \stopsolutions
7712 \fi

```

exnote

```

7713 \bool_if:NTF \c__problems_notes_bool {
7714 \newenvironment{exnote}[1][{}{
7715 \par\smallskip\hrule\smallskip
7716 \noindent\textbf{\prob@note@kw :~ }\small
7717 }{
7718 \smallskip\hrule
7719 }
7720 }{
7721 \excludecomment{exnote}
7722 }

```

hint

```

7723 \bool_if:NTF \c__problems_notes_bool {
7724 \newenvironment{hint}[1][{}{
7725 \par\smallskip\hrule\smallskip

```



```

7726 \noindent\textbf{\prob@hint@kw :~ }\small
7727 }{
7728 \smallskip\hrule
7729 }
7730 \newenvironment{exhint}[1][]{
7731 \par\smallskip\hrule\smallskip
7732 \noindent\textbf{\prob@hint@kw :~ }\small
7733 }{
7734 \smallskip\hrule
7735 }
7736 }{
7737 \excludecomment{hint}
7738 \excludecomment{exhint}
7739 }

```

gnote

```

7740 \bool_if:NTF \c__problems_notes_bool {
7741 \newenvironment{gnote}[1][]{
7742 \par\smallskip\hrule\smallskip
7743 \noindent\textbf{\prob@gnote@kw :~ }\small
7744 }{
7745 \smallskip\hrule
7746 }
7747 }{
7748 \excludecomment{gnote}
7749 }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7750 \newenvironment{mcb}{
7751 \begin{enumerate}
7752 }{
7753 \end{enumerate}
7754 }

```

we define the keys for the mcb macro

```

7755 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7756 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7757 \bool_set_true:N #1
7758 }{
7759 \bool_set_false:N #1
7760 }
7761 }
7762 \keys_define:nn { problem / mcb }{
7763 id .str_set_x:N = \l__problems_mcc_id_str ,
7764 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7765 T .default:n = { false } ,
7766 T .bool_set:N = \l__problems_mcc_t_bool ,
7767 F .default:n = { false } ,
7768 F .bool_set:N = \l__problems_mcc_f_bool ,

```

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

```

7769 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7770 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7771 }
7772 \cs_new_protected:Nn \l__problems_mcc_args:n {
7773   \str_clear:N \l__problems_mcc_id_str
7774   \tl_clear:N \l__problems_mcc_feedback_tl
7775   \bool_set_false:N \l__problems_mcc_t_bool
7776   \bool_set_false:N \l__problems_mcc_f_bool
7777   \tl_clear:N \l__problems_mcc_Ttext_tl
7778   \tl_clear:N \l__problems_mcc_Ftext_tl
7779   \str_clear:N \l__problems_mcc_id_str
7780   \keys_set:nn { problem / mcc }{ #1 }
7781 }

```

\mcc

```

7782 \def\mccTrueText{\textbf{(true)~}}
7783 \def\mccFalseText{\textbf{(false)~}}
7784 \newcommand\mcc[2][] {
7785   \l__problems_mcc_args:n{ #1 }
7786   \item[{$\Box$}] #2
7787   \ifsolutions
7788     \\\
7789     \bool_if:NT \l__problems_mcc_t_bool {
7790       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7791     }
7792     \bool_if:NT \l__problems_mcc_f_bool {
7793       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7794     }
7795     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7796       \emph{(\l__problems_mcc_feedback_tl)}
7797     }
7798   \fi
7799 } %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.

```

7800
7801 \keys_define:nn{ problem / inclproblem }{
7802   id      .str_set_x:N      = \l__problems_inclprob_id_str,
7803   pts     .tl_set:N        = \l__problems_inclprob_pts_tl,
7804   min     .tl_set:N        = \l__problems_inclprob_min_tl,
7805   title   .tl_set:N        = \l__problems_inclprob_title_tl,
7806   refnum  .int_set:N        = \l__problems_inclprob_refnum_int,
7807   type    .tl_set:N        = \l__problems_inclprob_type_tl,
7808   mhrepos .str_set_x:N      = \l__problems_inclprob_mhrepos_str
7809 }
7810 \cs_new_protected:Nn \__problems_inclprob_args:n {
7811   \str_clear:N \l__problems_prob_id_str

```

```

7812 \tl_clear:N \l__problems_inclprob_pts_tl
7813 \tl_clear:N \l__problems_inclprob_min_tl
7814 \tl_clear:N \l__problems_inclprob_title_tl
7815 \tl_clear:N \l__problems_inclprob_type_tl
7816 \int_zero_new:N \l__problems_inclprob_refnum_int
7817 \str_clear:N \l__problems_inclprob_mhrepos_str
7818 \keys_set:nn { problem / inclproblem }{ #1 }
7819 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7820   \let\l__problems_inclprob_pts_tl\undefined
7821 }
7822 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7823   \let\l__problems_inclprob_min_tl\undefined
7824 }
7825 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7826   \let\l__problems_inclprob_title_tl\undefined
7827 }
7828 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7829   \let\l__problems_inclprob_type_tl\undefined
7830 }
7831 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7832   \let\l__problems_inclprob_refnum_int\undefined
7833 }
7834 }
7835
7836 \cs_new_protected:Nn \__problems_inclprob_clear: {
7837   \let\l__problems_inclprob_id_str\undefined
7838   \let\l__problems_inclprob_pts_tl\undefined
7839   \let\l__problems_inclprob_min_tl\undefined
7840   \let\l__problems_inclprob_title_tl\undefined
7841   \let\l__problems_inclprob_type_tl\undefined
7842   \let\l__problems_inclprob_refnum_int\undefined
7843   \let\l__problems_inclprob_mhrepos_str\undefined
7844 }
7845 \__problems_inclprob_clear:
7846
7847 \newcommand\includeproblem[2][ ]{
7848   \__problems_inclprob_args:n{ #1 }
7849   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7850     \stex_html_backend:TF {
7851       \str_clear:N \l_tmpa_str
7852       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7853         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7854       }
7855       \stex_annotate_invisible:nnn{includeproblem}{
7856         \l_tmpa_str / #2
7857       }{}
7858     }{
7859       \begingroup
7860         \inputreftrue
7861         \tl_if_empty:nTF{ ##1 }{
7862           \input{#2}
7863         }{
7864           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7865         }

```

```

7866     \endgroup
7867   }
7868 }
7869 \__problems_inclprob_clear:
7870 }

```

(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.

```

7871 \AddToHook{enddocument}{
7872   \bool_if:NT \c__problems_pts_bool {
7873     \message{Total:~\arabic{pts}~points}
7874   }
7875   \bool_if:NT \c__problems_min_bool {
7876     \message{Total:~\arabic{min}~minutes}
7877   }
7878 }

```

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

```

7879 \def\pts#1{
7880   \bool_if:NT \c__problems_pts_bool {
7881     \marginpar{#1~\prob@pt@kw}
7882   }
7883 }
7884 \def\min#1{
7885   \bool_if:NT \c__problems_min_bool {
7886     \marginpar{#1~\prob@min@kw}
7887   }
7888 }

```

`\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.

```

7889 \newcounter{pts}
7890 \def\show@pts{
7891   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7892     \bool_if:NT \c__problems_pts_bool {
7893       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7894       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7895     }
7896   }{
7897     \tl_if_exist:NT \l__problems_prob_pts_tl {
7898       \bool_if:NT \c__problems_pts_bool {
7899         \tl_if_empty:NT\l__problems_prob_pts_tl{
7900           \tl_set:Nn \l__problems_prob_pts_tl {0}
7901         }
7902         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7903         \addtocounter{pts}{\l__problems_prob_pts_tl}
7904       }
7905     }

```

```

7906 }
7907 }

```

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

\show@min

```

7908 \newcounter{min}
7909 \def\show@min{
7910   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7911     \bool_if:NT \c__problems_min_bool {
7912       \marginpar{\l__problems_inclprob_pts_tl\ min}
7913       \addtocounter{min}{\l__problems_inclprob_min_tl}
7914     }
7915   }{
7916     \tl_if_exist:NT \l__problems_prob_min_tl {
7917       \bool_if:NT \c__problems_min_bool {
7918         \tl_if_empty:NT\l__problems_prob_min_tl{
7919           \tl_set:Nn \l__problems_prob_min_tl {0}
7920         }
7921         \marginpar{\l__problems_prob_min_tl\ min}
7922         \addtocounter{min}{\l__problems_prob_min_tl}
7923       }
7924     }
7925   }
7926 }
7927 \</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.

```
7928 \*package>
7929 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7930 \RequirePackage{13keys2e}
7931
7932 \newif\iftest\testfalse
7933 \DeclareOption{test}{\testtrue}
7934 \newif\ifmultiple\multiplefalse
7935 \DeclareOption{multiple}{\multipletrue}
7936 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7937 \ProcessOptions
```

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

```
7938 \RequirePackage{keyval}[1997/11/10]
7939 \RequirePackage{problem}
```

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

```
7940 \newcommand\hwexam@assignment@kw{Assignment}
7941 \newcommand\hwexam@given@kw{Given}
7942 \newcommand\hwexam@due@kw{Due}
7943 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7944 blank~for~extra~space}
7945 \def\hwexam@minutes@kw{minutes}
7946 \newcommand\correction@probs@kw{prob.}
7947 \newcommand\correction@pts@kw{total}
7948 \newcommand\correction@reached@kw{reached}
7949 \newcommand\correction@sum@kw{Sum}
7950 \newcommand\correction@grade@kw{grade}
7951 \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

```

7952 \AddToHook{begindocument}{
7953 \ltx@ifpackageloaded{babel}{
7954 \makeatletter
7955 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7956 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7957 \input{hwexam-ngerman.ldf}
7958 }
7959 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7960 \input{hwexam-finnish.ldf}
7961 }
7962 \clist_if_in:NnT \l_tmpa_clist {french}{
7963 \input{hwexam-french.ldf}
7964 }
7965 \clist_if_in:NnT \l_tmpa_clist {russian}{
7966 \input{hwexam-russian.ldf}
7967 }
7968 \makeatother
7969 }{}
7970 }
7971

```

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.

```

7972 \newcounter{assignment}
7973 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7974 \keys_define:nn { hwexam / assignment } {
7975 id .str_set:N = \l_@@_assign_id_str,
7976 number .int_set:N = \l_@@_assign_number_int,
7977 title .tl_set:N = \l_@@_assign_title_tl,
7978 type .tl_set:N = \l_@@_assign_type_tl,
7979 given .tl_set:N = \l_@@_assign_given_tl,
7980 due .tl_set:N = \l_@@_assign_due_tl,
7981 loadmodules .code:n = {
7982 \bool_set_true:N \l_@@_assign_loadmodules_bool
7983 }
7984 }
7985 \cs_new_protected:Nn \_@@_assignment_args:n {
7986 \str_clear:N \l_@@_assign_id_str
7987 \int_set:Nn \l_@@_assign_number_int {-1}
7988 \tl_clear:N \l_@@_assign_title_tl
7989 \tl_clear:N \l_@@_assign_type_tl
7990 \tl_clear:N \l_@@_assign_given_tl
7991 \tl_clear:N \l_@@_assign_due_tl
7992 \bool_set_false:N \l_@@_assign_loadmodules_bool
7993 \keys_set:nn { hwexam / assignment }{ #1 }
7994 }

```

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.

```

7995 \newcommand\given@due[2]{
7996 \bool_lazy_all:nF {
7997 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7998 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7999 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8000 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8001 }{ #1 }
8002
8003 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8004 \tl_if_empty:NF \l_@@_assign_given_tl {
8005 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8006 }
8007 }{
8008 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8009 }
8010
8011 \bool_lazy_or:nnF {
8012 \bool_lazy_and_p:nn {
8013 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8014 }{
8015 \tl_if_empty_p:V \l_@@_assign_due_tl
8016 }
8017 }{
8018 \bool_lazy_and_p:nn {
8019 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8020 }{
8021 \tl_if_empty_p:V \l_@@_assign_due_tl
8022 }
8023 }{ ,~ }
8024
8025 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8026 \tl_if_empty:NF \l_@@_assign_due_tl {
8027 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8028 }
8029 }{
8030 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8031 }
8032
8033 \bool_lazy_all:nF {
8034 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8035 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8036 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8037 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8038 }{ #2 }
8039 }

```

`\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.

```

8040 \newcommand\assignment@title[3]{
8041 \tl_if_empty:NTF \l_@@_inclassassign_title_tl {
8042 \tl_if_empty:NTF \l_@@_assign_title_tl {
8043 #1
8044 }{
8045 #2\l_@@_assign_title_tl#3
8046 }
8047 }{
8048 #2\l_@@_inclassassign_title_tl#3
8049 }
8050 }

```

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

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

```

8051 \newcommand\assignment@number{
8052 \int_compare:nNnTF \l_@@_inclassassign_number_int = {-1} {
8053 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8054 \arabic{assignment}
8055 } {
8056 \int_use:N \l_@@_assign_number_int
8057 }
8058 }{
8059 \int_use:N \l_@@_inclassassign_number_int
8060 }
8061 }

```

(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.

```

8062 \newenvironment{assignment}[1][]{
8063 \_@@_assignment_args:n { #1 }
8064 %\sref@target
8065 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8066 \global\stepcounter{assignment}
8067 }{
8068 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8069 }
8070 \setcounter{problem}{0}
8071 \renewcommand\prob@label[1]{\assignment@number.##1}
8072 \def\current@section@level{\document@hwexamtype}
8073 %\sref@label{id{\document@hwexamtype \thesection}
8074 \begin{@assignment}
8075 }{
8076 \end{@assignment}
8077 }

```

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

```

8078 \def\ass@title{
8079 {\protect\document@hwexamtype}\arabic{assignment}
8080 \assignment@title{}\;{}{}\;} -- \given@due{}\;{}
8081 }
8082 \ifmultiple
8083 \newenvironment{@assignment}{
8084 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8085 \begin{sfragment}[loadmodules]{\ass@title}
8086 }{
8087 \begin{sfragment}{\ass@title}
8088 }
8089 }{
8090 \end{sfragment}
8091 }

```

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

```

8092 \else
8093 \newenvironment{@assignment}{
8094 \begin{center}\bf
8095 \Large@title\strut\
8096 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}
8097 \large\given@due{--\;{}{}\;{};--}
8098 \end{center}
8099 }{}
8100 \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.

```

8101 \keys_define:nn { hwexam / inclassignment } {
8102 %id .str_set_x:N = \l_@@_assign_id_str,
8103 number .int_set:N = \l_@@_inclassign_number_int,
8104 title .tl_set:N = \l_@@_inclassign_title_tl,
8105 type .tl_set:N = \l_@@_inclassign_type_tl,
8106 given .tl_set:N = \l_@@_inclassign_given_tl,
8107 due .tl_set:N = \l_@@_inclassign_due_tl,
8108 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8109 }
8110 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8111 \int_set:Nn \l_@@_inclassign_number_int {-1}
8112 \tl_clear:N \l_@@_inclassign_title_tl
8113 \tl_clear:N \l_@@_inclassign_type_tl
8114 \tl_clear:N \l_@@_inclassign_given_tl
8115 \tl_clear:N \l_@@_inclassign_due_tl
8116 \str_clear:N \l_@@_inclassign_mhrepos_str
8117 \keys_set:nn { hwexam / inclassignment }{ #1 }
8118 }
8119 \l_@@_inclassignment_args:n {}
8120
8121 \newcommand\inputassignment[2][{}]{

```

```

8122 \_@@_inclassassignment_args:n { #1 }
8123 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8124 \input{#2}
8125 }{
8126 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8127 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8128 }
8129 }
8130 \_@@_inclassassignment_args:n {}
8131 }
8132 \newcommand\includeassignment[2][]{
8133 \newpage
8134 \inputassignment[#1]{#2}
8135 }

```

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

40.4 Typesetting Exams

\quizheading

```

8136 \ExplSyntaxOff
8137 \newcommand\quizheading[1]{%
8138 \def\@tas{#1}%
8139 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8140 \ifx\@tas\@empty\else%
8141 \noindent TA:~\@for\@I:=\@tas\do{{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8142 \fi%
8143 }
8144 \ExplSyntaxOn

```

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

\testheading

```

8145
8146 \def\hwexamheader{\input{hwexam-default.header}}
8147
8148 \def\hwexamminutes{
8149 \tl_if_empty:NTF \testheading@duration {
8150 {\testheading@min}~\hwexam@minutes@kw
8151 }{
8152 \testheading@duration
8153 }
8154 }
8155
8156 \keys_define:nn { hwexam / testheading } {
8157 min .tl_set:N = \testheading@min,
8158 duration .tl_set:N = \testheading@duration,
8159 reqpts .tl_set:N = \testheading@reqpts,
8160 tools .tl_set:N = \testheading@tools
8161 }
8162 \cs_new_protected:Nn \_@@_testheading_args:n {
8163 \tl_clear:N \testheading@min
8164 \tl_clear:N \testheading@duration

```

```

8165 \tl_clear:N \testheading@reqpts
8166 \tl_clear:N \testheading@tools
8167 \keys_set:nn { hwexam / testheading }{ #1 }
8168 }
8169 \newenvironment{testheading}[1][]{
8170 \_@@_testheading_args:n{ #1 }
8171 \newcount\check@time\check@time=\testheading@min
8172 \advance\check@time by -\theassignment@totalmin
8173 \newif\if@bonuspoints
8174 \tl_if_empty:NTF \testheading@reqpts {
8175 \@bonuspointsfalse
8176 }{
8177 \newcount\bonus@pts
8178 \bonus@pts=\theassignment@totalpts
8179 \advance\bonus@pts by -\testheading@reqpts
8180 \edef\bonus@pts{\the\bonus@pts}
8181 \@bonuspointstrue
8182 }
8183 \edef\check@time{\the\check@time}
8184
8185 \makeatletter\hwexamheader\makeatother
8186 }{
8187 \newpage
8188 }

```

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

\testspace

```

8189 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8190 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8191 \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.

```

8192 <@@=problems>
8193 \renewcommand\@problem[3]{
8194 \stepcounter{assignment@probs}
8195 \def\__problemspts{#2}
8196 \ifx\__problemspts\@empty\else
8197 \addtocounter{assignment@totalpts}{#2}
8198 \fi
8199 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8200 \xdef\correction@probs{\correction@probs & #1}%
8201 \xdef\correction@pts{\correction@pts & #2}
8202 \xdef\correction@reached{\correction@reached &}

```

```

8203 }
8204 <@@=hwexam>

```

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

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

```

8205 \newcounter{assignment@probs}
8206 \newcounter{assignment@totalpts}
8207 \newcounter{assignment@totalmin}
8208 \def\correction@probs{\correction@probs@kw}
8209 \def\correction@pts{\correction@pts@kw}
8210 \def\correction@reached{\correction@reached@kw}
8211 \stepcounter{assignment@probs}
8212 \newcommand\correction@table{
8213 \resizebox{\textwidth}{!}{%
8214 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8215 &\multicolumn{\theassignment@probs}{c|}||%|
8216 {\footnotesize\correction@forgrading@kw} &\\ \hline
8217 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8218 \correction@pts & \theassignment@totalpts & \\ \hline
8219 \correction@reached & & \[.7cm]\hline
8220 \end{tabular}}
8221 </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).