

The sTeX3 Package Collection *

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

2022-04-26

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.0 (last revised 2022-04-26)

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	101
25.1	Generic Path Handling	101
25.2	PWD and <code>kpsewhich</code>	103
25.3	File Hooks and Tracking	104
25.4	MathHub Repositories	105
25.5	Using Content in Archives	110
26	<code>sTeX</code>-References Implementation	114
26.1	Document URIs and URLs	114
26.2	Setting Reference Targets	116
26.3	Using References	118
27	<code>sTeX</code>-Modules Implementation	121
27.1	The <code>smodule</code> environment	125
27.2	Invoking modules	131

28	STEX-Module Inheritance Implementation	133
28.1	SMS Mode	133
28.2	Inheritance	137
29	STEX-Symbols Implementation	142
29.1	Symbol Declarations	142
29.2	Notations	149
29.3	Variables	158
30	STEX-Terms Implementation	164
30.1	Symbol Invocations	164
30.2	Terms	171
30.3	Notation Components	175
30.4	Variables	177
30.5	Sequences	179
31	STEX-Structural Features Implementation	180
31.1	Imports with modification	181
31.2	The feature environment	189
31.3	Structure	189
32	STEX-Statements Implementation	199
32.1	Definitions	199
32.2	Assertions	204
32.3	Examples	208
32.4	Logical Paragraphs	210
33	The Implementation	216
33.1	Proofs	216
33.2	Justifications	227
34	STEX-Others Implementation	228
35	STEX-Metattheory Implementation	229
36	Tikzinput Implementation	232
37	document-structure.sty Implementation	235
37.1	Package Options	235
37.2	Document Structure	236
37.3	Front and Backmatter	240
37.4	Global Variables	242
38	NotesSlides – Implementation	243
38.1	Class and Package Options	243
38.2	Notes and Slides	245
38.3	Header and Footer Lines	249
38.4	Frame Images	251
38.5	Colors and Highlighting	252
38.6	Sectioning	253
38.7	Excursions	255

39 The Implementation	257
39.1 Package Options	257
39.2 Problems and Solutions	258
39.3 Multiple Choice Blocks	265
39.4 Including Problems	266
39.5 Reporting Metadata	268
40 Implementation: The hwexam Package	270
40.1 Package Options	270
40.2 Assignments	271
40.3 Including Assignments	274
40.4 Typesetting Exams	275
40.5 Leftovers	277
41 References	278

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 $\text{\texttt{STeX}}$ 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 $\text{\texttt{LATeX}}$ more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

Both are legitimate and useful. The first requires a significantly smaller tool-chain, so we describe it first. The second requires a much more substantial (and experimental) toolchain of knowledge management systems. Both workflows profit from an integrated development environment (IDE), which (also) automates setup as far as possible (see [subsection 2.1.4](#)).

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

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

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

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

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

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

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

2.1.3 sTeX Archives (Manual Setup)

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

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/smgglom/<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 smgglom` will download all `smgglom` archives.

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

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

2.2 A First sTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

Note that all of the highlighting, tooltips, coloring and the environment headers come from `stexthm` – by default, the amount of additional packages loaded is kept to a minimum and all the presentations can be customized, see [chapter 6](#).

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

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

smodule First, we open a new *module* called `GeometricSeries`. The main purpose of the `smodule` environment is to group the contents and associate it with a *globally unique* identifier (URI), which is computed from the name `GeometricSeries` and the document context. (Depending on your pdf viewer), the URI should pop up in a tooltip if you hover over the word **geometric series**.

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

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

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

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

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

\symdef Next, we introduce a new *symbol* with name `geometric-series` and assign it the semantic macro `\geometricSeries`. `\symdef` also immediately assigns this symbol a *notation*, namely `S`.

\comp The macro `\comp` marks the `S` in the notation as a *notational component*, as opposed to e.g. arguments to `\geometricSeries`. It is the notational components that get highlighted and associated with the corresponding symbol (i.e. in this case `geometricSeries`). Since `\geometricSeries` takes no arguments, we can wrap the whole notation in a `\comp`.

```

\begin{sdefinition}[for=geometricSeries]
...
\end{sdefinition}
\begin{sassertion}[name=geometricSeriesConverges,type=theorem]
...
\end{sassertion}

```

What follows are two \LaTeX -statements (e.g. definitions, theorems, examples, proofs, ...). These are semantically marked-up variants of the usual environments, which take additional optional arguments (e.g. `for=`, `type=`, `name=`). Since many \LaTeX templates predefine environments like `definition` or `theorem` with different syntax, we use `sdefinition`, `sassertion`, `sexample` etc. instead. You can customize these environments to e.g. simply wrap around some predefined `theorem`-environment. That way, we can still use `sassertion` to provide semantic information, while being fully compatible with (and using the document presentation of) predefined environments.

In our case, the `stexthm`-package patches e.g. `\begin{sassertion}[type=theorem]` to use a `theorem`-environment defined (as usual) using the `amsthm` package.

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

`\symname`

The `\symname`-command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol.

Note that the argument of `\symref` can be a local or imported symbol (here the `series` symbol is imported from the `series` module). \LaTeX tries to determine the full symbol URI from the argument. If there are name clashes in or with the imported symbols, the name of the exporting module can be prepended to the symbol name before the `?` character.

If you hover over the word `series` in the pdf output, you should see a tooltip showing the full URI of the symbol used.

`\symref`

The `\symname`-command is a special case of the more general `\symref`-command, which allows customizing the precise text associated with a symbol. `\symref` takes two arguments the first is the symbol name, and the second a variant verbalization of the symbol, e.g. an inflection variant, a different language or a synonym. In our example `\symname{?series}` abbreviates `\symref{?series}{series}`.

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

`\definame`
`\definiendum`

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

```

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

```

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

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

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

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

2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then \TeX yields pretty colors and tooltips¹. But \TeX becomes a lot more powerful if we additionally convert our document to `xhtml` while preserving all the \TeX markup in the result.

TODO VSCode Plugin

Using `Ru\TeX` [RT], we can convert the document to `xhtml` using the command `rustex -i /path/to/file.tex -o /path/to/outfile.xhtml`. Investigating the resulting file, we notice additional semantic information resulting from our usage of semantic macros, `\symref` etc. Below is the (abbreviated) snippet inside our `\definiens` block:

```
<mrow resource="" property="stex:definiens">
  <mrow resource="...?series?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/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

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

Output:

Hello World

`\stexpatchmodule`

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

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

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

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

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

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

Output:

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

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

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

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

$\backslash\text{setnotation}$

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

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

3.3.3 Argument Modes

The notations so far used *simple* arguments which we call *mode-i* arguments. Declaring a new symbol with $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=3]$ is equivalent to writing $\backslash\text{symdecl}\{\text{foo}\}[\text{args}=iii]$, indicating that the semantic macro takes three mode-i arguments. However, there are three more argument modes which we will investigate now, namely mode-b, mode-a and mode-B arguments.

Mode-b Arguments

A mode-b argument represents a *variable* that is *bound* by the symbol in its application, making the symbol a *binding operator*. Typical examples of binding operators are e.g. sums \sum , products \prod , integrals \int , quantifiers like \forall and \exists , that λ -operator, etc.

\hookrightarrow Mode-b arguments behave exactly like mode-i arguments within T_EX, but applications of binding operators, i.e. symbols with mode-b arguments, are translated to OMBIND-terms in OMDoc/MMT, rather than OMA.

For example, we can implement a summation operator binding an index variable and taking lower and upper index bounds and the expression to sum over like this:

Example 9

Input:

```
1 \symdef{summation}[args=biii]
2   {\mathop{\comp{\sum}}_{\#1\comp{=}\#2}^{\#3}\#4}
3   $\summation{\svar{x}}{1}{\svar{n}}{\svar{x}}^2$
```

Output:

$$\sum_{x=1}^n x^2$$

where the variable x is now *bound* by the `\summation`-symbol in the expression.

Mode-a Arguments

Mode-a arguments represent a *flexary argument sequence*, i.e. a sequence of arguments of arbitrary length. Formally, operators that take arbitrarily many arguments don't "exist", but in informal mathematics, they are ubiquitous. Mode-a arguments allow us to write e.g. `\addition{a,b,c,d,e}` rather than having to write something like `\addition{a}{\addition{b}{\addition{c}{\addition{d}{e}}}}`!

`\notation` (and consequently `\symdef`, too) take one additional argument for each mode-a argument that indicates how to "accumulate" a comma-separated sequence of arguments. This is best demonstrated on an example.

Let's say we want an operator representing quantification over an ascending chain of elements in some set, i.e. `\ascendingchain{S}{a,b,c,d,e}{t}` should yield $\forall a <_S b <_S c <_S d <_S e. t$. The "base"-notation for this operator is simply `{\comp{\forall} \#2 \comp{. , } \#3}`, where `\#2` represents the full notation fragment *accumulated* from `{a,b,c,d,e}`.

The *additional* argument to `\notation` (or `\symdef`) takes the same arguments as the base notation and two *additional* arguments `\#1` and `\#2` representing successive pairs in the mode-a argument, and accumulates them into `\#2`, i.e. to produce $a <_S b <_S c <_S d <_S e$, we do `{\#1 \comp{<}_{\#1} \#2}`:

Example 10

Input:

```

1 \symdef{ascendingchain}[args=iai]
2   {\comp{\forall} #2\comp{. \,} #3}
3   {##1 \comp{<}_{#1} ##2}
4
5 Tadaa: $\ascendingchain{S}{a,b,c,d,e}{t}$

```

Output:

Tadaa: $\forall a <_S b <_S c <_S d <_S e. t$

If this seems overkill, keep in mind that you will rarely need the single-hash arguments #1,#2 etc. in the a-notation-argument. For a much more representative and simpler example, we can introduce flexary addition via:

Example 11

Input:

```

1 \symdef{addition}[args=a]{#1}{##1 \comp{+} ##2}
2
3 Tadaa: $\addition{a,b,c,d,e}$

```

Output:

Tadaa: $a+b+c+d+e$

The assoc-key We mentioned earlier that “formally”, flexary arguments don’t really “exist”. Indeed, formally, addition is usually defined as a binary operation, quantifiers bind a single variable etc.

Consequently, we can tell \LaTeX (or, rather, MMT/OMDOC) how to “resolve” flexary arguments by providing `\symdecl` or `\symdef` with an optional `assoc`-argument, as in `\symdecl{addition}[args=a,assoc=bin]`. The possible values for the `assoc`-key are:

bin: A binary, associative argument, e.g. as in `\addition`

binl: A binary, left-associative argument, e.g. $a^{b^{c^d}}$, which stands for $((a^b)^c)^d$

binr: A binary, right-associative argument, e.g. as in $A \rightarrow B \rightarrow C \rightarrow D$, which stands for $A \rightarrow (B \rightarrow (C \rightarrow D))$

pre: Successively prefixed, e.g. as in $\forall x, y, z. P$, which stands for $\forall x. \forall y. \forall z. P$

conj: Conjunctive, e.g. as in $a = b = c = d$ or $a, b, c, d \in A$, which stand for $a = d \wedge b = d \wedge c = d$ and $a \in A \wedge b \in A \wedge c \in A \wedge d \in A$, respectively

pwconj: Pairwise conjunctive, e.g. as in $a \neq b \neq c \neq d$, which stands for $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

As before, at the PDF level, this annotation is invisible (and without effect), but at the level of the generated OMDoc/MMT this leads to more semantical expressions.

Mode-B Arguments

Finally, mode-B arguments simply combine the functionality of both `a` and `b` - i.e. they represent an arbitrarily long sequence of variables to be bound, e.g. for implementing quantifiers:

Example 12

Input:

```
1 \symdef{quantforall}[args=Bi]
2   {\comp{\forall}#1\comp{.}#2}
3   {##1\comp,##2}
4
5 \$\quantforall{\svar{x},\svar{y},\svar{z}}{P}$
```

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

`\symdecl` and `\symdef` take two more optional arguments. \TeX largely ignores them (except for special situations we will talk about later), but MMT can pick up on them for additional services. These are the `type` and `def` keys, which expect expressions in math-mode (ideally using semantic macros, of course!)

The `type` and `def` keys correspond to the `type` and `definiens` components of

- \hookrightarrow OMDOC/MMT constants.
- \hookrightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \hookrightarrow OMDOC/MMT – being foundation-independent – does not a priori implement a fixed typing system.

The `type`-key allows us to provide additional information (given the necessary \TeX symbols), e.g. for addition on natural numbers:

Example 13

Input:

```
1 \symdef{Nat}[type=\set]{\comp{\mathbb N}}
2 \symdef{addition}[
3   type=\funtype{\Nat,\Nat}{\Nat},
4   op=+,
5   args=a
6 ]{#1}{##1 \comp+ ##2}
7
8 \symname{addition} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

`addition` is an operation $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

The `def`-key allows for declaring symbols as abbreviations:

Example 14

Input:

```
1 \symdef{successor}[
2   type=\funtype{\Nat}{\Nat},
3   def=\fun{\svar{x}}{\addition{\svar{x},1}},
4   op=\mathtt{succ},
5   args=1
6 ]{\comp{\mathtt{succ}{}#1\comp{}}}
7
8 The \symname{successor} operation $\funtype{\Nat}{\Nat}$
9 is defined as $\fun{\svar{x}}{\addition{\svar{x},1}}$
```

Output:

The `successor` operation $\mathbb{N} \rightarrow \mathbb{N}$ is defined as $x \mapsto x+1$

3.3.5 Precedences and Automated Bracketing

Having done `\addition`, the obvious next thing to implement is `\multiplication`. This is straight-forward in theory:

Example 15

Input:

```
1 \symdef{multiplication}[
2   type=\funtype{\Nat,\Nat}{\Nat},
3   op=\cdot,
4   args=a
5 ]{\#1}{\#1 \comp\cdot \#2}
6
7 \symname{multiplication} is an operation $\funtype{\Nat,\Nat}{\Nat}$
```

Output:

`multiplication` is an operation $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$

However, if we *combine* `\addition` and `\multiplication`, we notice a problem:

Example 16

Input:

```
1 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$a+b \cdot c+d \cdot e$

We all know that \cdot binds stronger than $+$, so the output $a+b\cdot c+d\cdot e$ does not actually reflect the term we wrote. We can of course insert parentheses manually

Example 17

Input:

```
1 $\addition{a,\multiplication{b,(\addition{c,\multiplication{d,e}})}}$
```

Output:

$$a+b\cdot(c+d\cdot e)$$

but we can also do better by supplying *precedences* and have \TeX insert parentheses automatically.

For that purpose, `\notation` (and hence `\symdef`) take an optional argument `prec=<opprec>;<argprec1>x...x<argprec n>`.

We will investigate the precise meaning of `<opprec>` and the `<argprec>`s shortly – in the vast majority of cases, it is perfectly sufficient to think of `prec=` taking a single number and having that be *the* precedence of the notation, where lower precedences (somewhat counterintuitively) bind stronger than higher precedences. So fixing our notations for `\addition` and `\multiplication`, we get:

Example 18

Input:

```
1 \notation{multiplication}[
2   op=\cdot,
3   prec=50
4 ]{#1}{##1 \comp\cdot ##2}
5 \notation{addition}[
6   op=+,
7   prec=100
8 ]{#1}{##1 \comp+ ##2}
9
10 $\addition{a,\multiplication{b,\addition{c,\multiplication{d,e}}}}$
```

Output:

$$a+b\cdot(c+d\cdot e)$$

Note that the precise numbers used for precedences are pretty arbitrary – what matters is which precedences are higher than which other precedences when used in conjunction.

`\infprec`
`\neginfprec`

It is occasionally useful to have “infinitely” high or low precedences to enforce or forbid automated bracketing entirely – for those purposes, `\infprec` and `\neginfprec` exist (which are implemented as the maximal and minimal integer values accordingly).

More precisely, each notation takes

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

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

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

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

In the example above:



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

3.3.6 Variables

All symbol and notation declarations require a module with which they are associated, hence the commands `\symdecl`, `\notation`, `\symdef` etc. are disabled outside of `smodule`-environments.

Variables are different – variables are allowed everywhere, are not exported when the current module (if one exists) is imported (via `\importmodule` or `\usemodule`) and (also unlike symbol declarations) “disappear” at the end of the current \TeX group.

`\svar`

So far, we have always used variables using `\svar{n}`, which marks-up n as a variable with name `n`. More generally, `\svar[foo]{<texcode>}` marks-up the arbitrary `<texcode>` as representing a variable with name `foo`.

Of course, this makes it difficult to reuse variables, or introduce “functional” variables with arities > 0 , or provide them with a type or definiens.

\vardef

For that, we can use the `\vardef` command. Its syntax is largely the same as that of `\symdef`, but unlike symbols, variables have only one notation (TODO: so far?), hence there is only `\vardef` and no `\vardecl`.

Example 19

Input:

```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfprec
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 $\text{\S}\text{\TeX}$ Metatheory)

The `stex-metatheory` package contains $\text{\S}\text{\TeX}$ 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 $\text{\S}\text{\TeX}$ 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 $\text{\S}\text{\TeX}$ 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 $\text{\S}\text{\TeX}$ collection rather than encoding it in $\text{\S}\text{\TeX}$ itself⁴

⁴EdNOTE: MK: why? continue

Chapter 4

Using \TeX Symbols

Given a symbol declaration `\symdecl{symbolname}`, we obtain a semantic macro `\symbolname`. We can use this semantic macro in math mode to use its notation(s), and we can use `\symbolname!` in math mode to use its operator notation(s). What else can we do?

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

Note that `\symname` uses the *name* of a symbol, not its macroname. More precisely, `\symname` will insert the name of the symbol with “-” replaced by spaces. If a symbol does not have an explicit `name=` given, the two are equal – but for `\symname` it often makes sense to make the two explicitly distinct. For example:

Example 32

Input:

```
1 \symdef{Nat}[
2   name=natural-number,
3   type=\set
4 ]{\comp{\mathbb{N}}}
5
6 A \symname{Nat} is...
```

Output:

A natural number is...

`\symname` takes two additional optional arguments, `pre=` and `post=` that get prepended or appended respectively to the symbol name.

`\Symname`

Additionally, `\Symname` behaves exactly like `\symname`, but will capitalize the first letter of the name:

Example 33

Input:

```
1 \Symname[post=s]{Nat} are...
```

Output:

Natural numbers are...



This is as good a place as any other to explain how \TeX resolves a string `symbolname` to an actual symbol.

If `\symbolname` is a semantic macro, then \TeX has no trouble resolving `symbolname` to the full URI of the symbol that is being invoked.

However, especially in `\symname` (or if a symbol was introduced using `\symdecl*` without generating a semantic macro), we might prefer to use the *name* of a symbol directly for readability – e.g. we would want to write `A \symname{natural-number} is...` rather than `A \symname{Nat} is...`. \TeX attempts to handle this case thusly:

If `string` does *not* correspond to a semantic macro `\string` and does *not* contain a `?`, then \TeX checks all symbols currently in scope until it finds one, whose name is `string`. If `string` is of the form `pre?name`, \TeX first looks through all modules currently in scope, whose full URI ends with `pre`, and then looks for a symbol with name `name` in those. This allows for disambiguating more precisely, e.g. by saying `\symname{Integers?addition}` or `\symname{RealNumbers?addition}` in the case where several `additions` are in scope.

4.2 Marking Up Text and On-the-Fly Notations

We can also use semantic macros outside of text mode though, which allows us to annotate arbitrary text fragments.

Let us assume again, that we have `\symdef{addition}[args=2]{#1 \comp+ #2}`. Then we can do

Example 34

Input:

```
1 \addition{\comp{The sum of} \arg{\$svar{n}}\$} \comp{ and } \arg{\$svar{m}}\$}
2 is...
```

Output:

The sum of n and m is...

...which marks up the text fragment as representing an *application* of the `addition`-symbol to two argument n and m .

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



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

\arg

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

Output:

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

adding k yields...

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

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

Example 37

Input:

```
1 Given $\addition{\svar{n}}{\svar{m}}$, then
2 $\addition*{
3   \arg*{\addition{\svar{n}}{\svar{m}}}
4   \comp{+}
5   \arg{\svar{k}}
6 }$ yields...
```

Output:

Given $n+m$, then $+k$ yields...

4.3 Referencing Symbols and Statements

TODO: references documentation

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

Chapter 5

sTEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

As mentioned earlier, we can semantically mark-up *statements* such as definitions, theorems, lemmata, examples, etc.

The corresponding environments for that are:

- `sdefinition` for definitions,
- `sassertion` for assertions, i.e. propositions that are declared to be *true*, such as theorems, lemmata, axioms,
- `sexample` for examples and counterexamples, and
- `sparagraph` for “other” semantic paragraphs, such as comments, remarks, conjectures, etc.

The *presentation* of these environments can be customized to use e.g. predefined theorem-environments, see [chapter 6](#) for details.

All of these environments take optional arguments in the form of `key=value`-pairs. Common to all of them are the keys `id=` (for cross-referencing, see [section 4.3](#)), `type=` for customization (see [chapter 6](#)) and additional information (e.g. definition principles, “difficulty” etc), as well as `title=` (for giving the paragraph a title), and finally `for=`.

The `for=` key expects a comma-separated list of existing symbols, allowing for e.g. things like

Example 38

Input:

```
1 \begin{sexample}[
2   id=additionandmultiplication.ex,
3   for={addition,multiplication},
4   type={trivial,boring},
5   title={An Example}
6 ]
7   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
8 \end{sexample}
```

Output:

Example 5.1.1 (An Example). $2+3$ is 5, $2\cdot 3$ is 6.

`\definiendum`
`\definame`
`\Definame`

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

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

`\definiens`

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

All four statement environments – i.e. **sdefinition**, **sassertion**, **sexample**, and **sparagraph** – also take an optional parameter `name=` – if this one is given a value, the environment will generate a *symbol* by that name (but with no semantic macro). Not only does this allow for `\symref` et al, it allows us to resume our earlier example for monoids much more nicely:⁷

Example 39

Input:

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

```

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

```

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

Axiom 5.1.3 (Unit). $x \circ e = x$ for all $x \in U$

An example for a **monoid** is...

.

The main difference to before⁸ is that the two **sassertions** now have **name=** attributes. Thus the **mathstructure monoid** now contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. \circ is associative, but *the assertion that it is actually true* that \circ is associative.

If we now want to instantiate **monoid** (unless with a variable, of course), we also need to assign **associative** and **neutral** to analogous assertions. So the earlier example

```

1 \instantiate{intmonoid}{monoid}{\mathbb{Z}_{+,0}}[
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 ]

```

⁸EDNOTE: MK: reference

...will not work anymore. We now need to give assertions that addition is associative and that zero is a unit with respect to addition.²

The `stex-proof` package supplies macros and environment that allow to annotate the structure of mathematical proofs in \LaTeX document. This structure can be used by MKM systems for added-value services, either directly from the \LaTeX sources, or after translation.

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

```

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

```

This yields the following result:

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

²Of course, \LaTeX can not check that the assertions are the “correct” ones – but if the assertions (both in monoid as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

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

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

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

1.3. $n > 1$:

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

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

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

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

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

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

spproof The **spproof** environment is the main container for proofs. It takes an optional **KeyVal** argument that allows to specify the **id** (identifier) and **for** (for which assertion is this a proof) keys. The regular argument of the **proof** environment contains an introductory comment, that may be used to announce the proof style. The **proof** environment contains a sequence of **spfststep**, **spfstcomment**, and **spfstcases** environments that are used to markup the proof steps.

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

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

spfststep Regular proof steps are marked up with the **step** environment, which takes an optional **KeyVal** argument for annotations. A proof step usually contains a local assertion (the text of the step) together with some kind of evidence that this can be derived from already established assertions.

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

`\sproofend`

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

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

`\sProofEndSymbol`

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

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

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

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

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

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

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

```

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

```

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

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

image

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

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

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

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

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

7.2 Modular Document Structuring

The `document-structure` package supplies an infrastructure for writing OMDOC documents in L^AT_EX. This includes a simple structure sharing mechanism for S_TE_X that allows to move from a copy-and-paste document development model to a copy-and-reference model, which conserves space and simplifies document management. The augmented structure can be used by MKM systems for added-value services, either directly from the S_TE_X sources, or after translation.

The `document-structure` package supplies macros and environments that allow to label document fragments and to reference them later in the same document or in other documents. In essence, this enhances the document-as-trees model to documents-as-directed-acyclic-graphs (DAG) model. This structure can be used by MKM systems for added-value services, either directly from the S_TE_X sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the S_TE_X collection.

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document source and the formatter does the copying during document formatting/presentation.

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

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

sfragment The structure of the document is given by nested `sfragment` environments. In the L^AT_EX route, the `sfragment` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `sfragment` environments. Correspondingly, the `sfragment` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the sfragment. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]. The option `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`¹⁰, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

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

```

¹⁰EdNOTE: MK: still?

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindfragment` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct.¹¹

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

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

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

`\prematurestop`
`\afterprematurestop`

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

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

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

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

7.3 Slides and Course Notes

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

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

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

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



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

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

```

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

```

`\ifnotes`

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

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



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



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

`\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

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

`\setlicensing`

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

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

`\frameimage`
`\mhframeimage`

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

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

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

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

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

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

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

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

`\textwarning`

The `\textwarning` macro generates a warning sign: 

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

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

`\excursion`

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

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

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

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

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

`\excursiongroup`

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

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



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

7.4 Representing Problems and Solutions

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

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

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

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

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

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

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

Example 40

Input:

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

`\ifsolutions`

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

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

`\mcc`

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

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

If we start the solutions, then we get

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

What is the keyword to introduce a function definition in python?

- ☐ `def`
(**true**)
- ☐ `function`
(**false**) (*that is for C and C++*)
- ☐ `fun`
(**false**) (*that is for Standard ML*)
- ☐ `public static void`
(**false**) (*that is for Java*)

¹²without solutions (that is what the students see during the exam/quiz)¹²

¹²EdNOTE: MK: that did not work!

Example 42

Input:

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

Output:

Problem 7.4.3 (Functions)

What is the keyword to introduce a function definition in python?

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

\includeproblem

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

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

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

7.5 Homeworks, Quizzes and Exams

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up

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

<code>solutions</code>	The <code>wexam</code> package and class take the options <code>solutions</code> , <code>notes</code> , <code>hints</code> , <code>gnotes</code> , <code>pts</code> , <code>min</code> , and <code>boxed</code> that are just passed on to the <code>problems</code> package (cf. its documentation for a description of the intended behavior).
<code>notes</code>	
<code>hints</code>	
<code>gnotes</code>	
<code>pts</code>	
<code>min</code>	
<code>assignment</code>	This package supplies the <code>assignment</code> environment that groups problems into assignment sheets. It takes an optional <code>KeyVal</code> argument with the keys <code>number</code> (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — the ordinal of the <code>assignment</code> environment), <code>title</code> (for the assignment title; this is referenced in the title of the assignment sheet), <code>type</code> (for the assignment type; e.g. “quiz”, or “homework”), <code>given</code> (for the date the assignment was given), and <code>due</code> (for the date the assignment is due).
<code>number</code>	
<code>title</code>	
<code>type</code>	
<code>given</code>	
<code>due</code>	
<code>multiple</code>	Furthermore, the <code>hwexam</code> package takes the option <code>multiple</code> that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).
<code>test</code>	Finally, there is the option <code>test</code> that modifies the behavior to facilitate formatting tests. Only in <code>test</code> mode, the macros <code>\testspace</code> , <code>\testnewpage</code> , and <code>\testemptypage</code> have an effect: they generate space for the students to solve the given problems. Thus they can be left in the \LaTeX source.
<code>\testspace</code>	<code>\testspace</code> takes an argument that expands to a dimension, and leaves vertical space accordingly. <code>\testnewpage</code> makes a new page in <code>test</code> mode, and <code>\testemptypage</code> generates an empty page with the cautionary message that this page was intentionally left empty.
<code>\testnewpage</code>	
<code>\testemptypage</code>	
<code>testheading</code>	Finally, the <code>\testheading</code> takes an optional keyword argument where the keys <code>duration</code> specifies a string that specifies the duration of the test, <code>min</code> specifies the equivalent in number of minutes, and <code>reqpts</code> the points that are required for a perfect grade.
<code>duration</code>	
<code>min</code>	
<code>reqpts</code>	
	<pre> 1 \title{320101 General Computer Science (Fall 2010)} 2 \begin{testheading}[duration=one hour,min=60,reqpts=27] 3 Good luck to all students! 4 \end{testheading} </pre>

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-04-26

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>mhrefpos</code> . It then resolves the file path in <code>\mhgraphics[mhrefpos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additionally 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:

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\stex_activate_module:n`

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists – all of which either declare module content or are needed in order to declare module content:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. sms mode continues immediately after. These macros may not take any arguments or otherwise gobble tokens.

Initially: `\makeatletter`, `\makeatother`, `\ExplSyntaxOn`, `\ExplSyntaxOff`.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed and potentially gobble up further tokens. These macros need to make sure, that the very last token they ultimately expand to is `\stex_smsmode_do:`.

Initially: `\symdecl`, `\notation`, `\symdef`, `\importmodule`, `\STEXexport`, `\inlineass`, `\inlinedef`, `\inlineex`, `\endinput`, `\setnotation`, `\copynotation`.

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_do:` needs to be the last token in the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called directly and sms mode continues afterwards.

Initially: `smodule`, `copymodule`, `interpretmodule`, `sdefinition`, `sexample`, `sassertion`, `sparagraph`.

`\stex_if_smsmode_p: *`
`\stex_if_smsmode: TF *`

Tests whether SMS mode is currently active.

<hr/> <hr/> <code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn</code> $\{\langle filename \rangle\}$ $\{\langle code \rangle\}$
	Executes $\langle code \rangle$ in SMS mode, followed by the content of $\langle filename \rangle$. $\langle code \rangle$ can be used e.g. to set the current repository, and is executed within a new tex group, and the same group as the file content.

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

12.1.2 Imports and Inheritance

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

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

\stex_import_module_uri:nn

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

- **name**: An (OMDOC) name. By default equal to $\langle\text{macroname}\rangle$.
- **type**: An (ideally semantic) term, representing a *type*. Not used by STEX, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i** a “normal” argument, e.g. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{ii}]$ allows for $\backslash\text{plus}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\text{symdecl}\{\text{plus}\}[\text{args}=\text{a}]$ allows for $\backslash\text{plus}\{2,2,2\}$.
 - b** a *variable* argument. Is treated by STEX like an *i*-argument, but an application is turned into an **OMBind** in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. $\backslash\text{symdecl}\{\text{forall}\}[\text{args}=\text{bi}]$ allows for $\backslash\text{forall}\{x\in\text{Nat}\}\{x\geq 0\}$.

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

<hr/> <hr/> <code>\STEXsymbol</code>	Uses <code>\stex_get_symbol:n</code> to find the symbol denoted by the first argument and passes the result on to <code>\stex_invoke_symbol:n</code>
<hr/> <hr/> <code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
<hr/> <hr/> <code>\stex_invoke_symbol:n</code>	Executes a semantic macro. Outside of math mode or if followed by <code>*</code> , it continues to <code>\stex_term_custom:nn</code> . In math mode, it uses the default or optionally provided notation of the associated symbol. If followed by <code>!</code> , it will invoke the symbol <i>itself</i> rather than its application (and continue to <code>\stex_term_custom:nn</code>), i.e. it allows to refer to <code>\plus!</code> [addition] as an operation, rather than <code>\plus[addition of]{some}{terms}</code> .
<hr/> <hr/> <code>_stex_term_math_oms:nnnn</code> <code>_stex_term_math_oma:nnnn</code> <code>_stex_term_math_omb:nnnn</code>	<code><URI><fragment><precedence><body></code> Annotates <code><body></code> as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol <code><URI></code> , generated by the specific notation <code><fragment></code> with (upwards) operator precedence <code><precedence></code> . Inserts parentheses according to the current downwards precedence and operator precedence.
<hr/> <hr/> <code>_stex_term_math_arg:nnn</code>	<code>\stex_term_arg:nnn<int><prec><body></code> Annotates <code><body></code> as the <code><int></code> th argument of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> .
<hr/> <hr/> <code>_stex_term_math_assoc_arg:nnnn</code>	<code>\stex_term_arg:nnn<int><prec><notation><body></code> Annotates <code><body></code> as the <code><int></code> th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence <code><prec></code> and associative notation <code><notation></code> .

<hr/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<code>\dobrackets {⟨body⟩}</code> Puts <i>⟨body⟩</i> in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S TEX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <code>\withbrackets</code> <hr/>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within <i>⟨body⟩</i>) sets the brackets used by \S TEX for automated bracketing (by default (and)) to <i>⟨left⟩</i> and <i>⟨right⟩</i> . Note that <i>⟨left⟩</i> and <i>⟨right⟩</i> need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <code>\stex_term_custom:nn</code> <hr/>	<code>\stex_term_custom:nn{⟨URI⟩}{⟨args⟩}</code> Implements custom one-time notation. Invoked by <code>\stex_invoke_symbol:n</code> in text mode, or if followed by <code>*</code> in math mode, or whenever followed by <code>!</code> .
<hr/> <code>\comp</code> <code>\compemph</code> <code>\compemph@uri</code> <code>\defemph</code> <code>\defemph@uri</code> <code>\symrefemph</code> <code>\symrefemph@uri</code> <code>\varemp</code> <code>\varemp@uri</code> <hr/>	<code>\comp{⟨args⟩}</code> Marks <i>⟨args⟩</i> as a notation component of the current symbol for highlighting, linking, etc. The precise behavior is governed by <code>\@comp</code> , which takes as additional argument the URI of the current symbol. By default, <code>\@comp</code> adds the URI as a PDF tooltip and colors the highlighted part in blue. <code>\@defemph</code> behaves like <code>\@comp</code> , and can be similarly redefined, but marks an expression as <i>definiendum</i> (used by <code>\definiendum</code>)
<hr/> <code>\STEXinvisible</code> <hr/>	Exports its argument as OMDoc (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.
<hr/> <code>\ellipses</code> <hr/>	TODO

Chapter 15

ST_EX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

`symboldoc` `\begin{<symboldoc>}{<symbols>} <text> \end{<symboldoc>}`
 Declares *<text>* to be a (natural language, encyclopaedic) description of *{<symbols>}*
 (a comma separated list of symbol identifiers).

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

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

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV

Implementation

Chapter 24

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

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

```

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

```

24.2 Preliminaries

```

45 \<*package>
46
47 %%%%%%%%% basics.dtx %%%%%%%%%
48
49 \RequirePackage{expl3,l3keys2e,ltxcmds}
50 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
51
52 \bool_if_exist:NF \c_stex_document_class_bool {
53   \bool_set_false:N \c_stex_document_class_bool
54   \RequirePackage{standalone}
55 }
56
57 \message{^^J
58   *****^^J
59   *~This~is~sTeX~version~3.1.0~*^^J
60   *****^^J
61 ^^J}
62
63 %\RequirePackage{morewrites}
64 %\RequirePackage{amsmath}
65
66 Package options:
67 \keys_define:nn { stex } {
68   debug      .clist_set:N = \c_stex_debug_clist ,
69   lang       .clist_set:N = \c_stex_languages_clist ,
70   mathhub    .tl_set_x:N  = \mathhub ,
71   usesms     .bool_set:N  = \c_stex_persist_mode_bool ,
72   writesms   .bool_set:N  = \c_stex_persist_write_mode_bool ,
73   image      .bool_set:N  = \c_tikzinput_image_bool ,
74   unknown    .code:n      = {}
75 }
76 \ProcessKeysOptions { stex }
77
78 \stex The sTeX logo:
79 \TeX
80 \RequirePackage{xspace}
81 \protected\def\stex{

```

```

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

```

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

24.3 Messages and logging

```

82 <@@=stex_log>
    Warnings and error messages
83 \msg_new:nnn{stex}{error/unknownlanguage}{
84   Unknown~language:~#1
85 }
86 \msg_new:nnn{stex}{warning/nomathhub}{
87   MATHHUB~system~variable~not~found~and~no~
88   \detokenize{\mathhub}~value~set!
89 }
90 \msg_new:nnn{stex}{error/deactivated-macro}{
91   The~\detokenize{#1}~command~is~only~allowed~in~#2!
92 }

```

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

```

93 \cs_new_protected:Nn \stex_debug:nn {
94   \clist_if_in:NnTF \c_stex_debug_clist { all } {
95     \msg_set:nnn{stex}{debug / #1}{
96       \\Debug~#1:~#2\\
97     }
98     \msg_none:nn{stex}{debug / #1}
99   }{
100     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
101       \msg_set:nnn{stex}{debug / #1}{
102         \\Debug~#1:~#2\\
103       }
104       \msg_none:nn{stex}{debug / #1}
105     }
106   }
107 }

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

117 `<@=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`

118 `\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`

```
119 \cs_new_protected:Nn \_stex_html_checkempty:n {
120   \tl_set:Nn \l_stex_html_arg_tl { #1 }
121   \tl_if_empty:NT \l_stex_html_arg_tl {
122     \tl_set_eq:NN \l_stex_html_arg_tl \c_stex_html_emptyarg_tl
123   }
124 }
```

(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`

```
125 \bool_new:N \_stex_html_do_output_bool
126 \bool_set_true:N \_stex_html_do_output_bool
127
128 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
129   \bool_if:nTF \_stex_html_do_output_bool
130     \prg_return_true: \prg_return_false:
131 }
```

(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

```
132 \cs_new_protected:Nn \stex_suppress_html:n {
133   \exp_args:Nne \use:nn {
134     \bool_set_false:N \_stex_html_do_output_bool
135     #1
136   }{
137     \stex_if_do_html:T {
138       \bool_set_true:N \_stex_html_do_output_bool
139     }
140   }
141 }
```

(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.

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

```

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

```

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

24.5 Babel Languages

```

154 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

```

```

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

```

24.6 Persistence

```

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

```

```

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

```

24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

261 \cs_new_protected:Nn \stex_reactivate_macro:N {
262   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
263 }

```

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

`\ignorespacesandpars`

```

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

```



```

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

```

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

`\MMTrule`

```

331 \NewDocumentCommand \MMTrule {m m}{
332   \seq_set_split:Nnn \l_tmpa_seq , {#2}
333   \int_zero:N \l_tmpa_int
334   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
335     \seq_if_empty:NF \l_tmpa_seq {
336       $\seq_map_inline:Nn \l_tmpa_seq {
337         \int_incr:N \l_tmpa_int
338         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
339       }$
340     }
341   }
342 }
343
344 \NewDocumentCommand \MMTinclude {m}{
345   \stex_annotate_invisible:nnn{import}{#1}{ }
346 }
347
348 \tl_new:N \g_stex_document_title
349 \cs_new_protected:Npn \STEXtitle #1 {
350   \tl_if_empty:NT \g_stex_document_title {
351     \tl_gset:Nn \g_stex_document_title { #1 }
352   }
353 }
354 \cs_new_protected:Nn \stex_document_title:n {
355   \tl_if_empty:NT \g_stex_document_title {
356     \tl_gset:Nn \g_stex_document_title { #1 }
357     \stex_annotate_invisible:n{\noindent
358       \stex_annotate:nnn{doctitle}{ }{ #1 }
359     \par}
360   }
361 }
362 \AtBeginDocument {
363   \let \STEXtitle \stex_document_title:n
364   \tl_if_empty:NF \g_stex_document_title {
365     \stex_annotate_invisible:n{\noindent
366       \stex_annotate:nnn{doctitle}{ }{ \g_stex_document_title }
367     \par}
368   }
369 }
370
371 </package>

```

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

Chapter 25

STEX -MathHub Implementation

```
372 <*package>
373
374 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
375
376 <@@=stex_path>
    Warnings and error messages
377 \msg_new:nnn{stex}{error/norepository}{
378   No~archive~#1~found~in~#2
379 }
380 \msg_new:nnn{stex}{error/notinarchive}{
381   Not~currently~in~an~archive,~but~\detokenize{#1}~
382   needs~one!
383 }
384 \msg_new:nnn{stex}{error/nofile}{
385   \detokenize{#1}~could~not~find~file~#2
386 }
387 \msg_new:nnn{stex}{error/twofiles}{
388   \detokenize{#1}~found~two~candidates~for~#2
389 }
```

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`

```
390 \cs_new_protected:Nn \stex_path_from_string:Nn {
391   \str_set:Nx \l_tmpa_str { #2 }
392   \str_if_empty:NTF \l_tmpa_str {
393     \seq_clear:N #1
394   }{
395     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
396     \sys_if_platform_windows:T{
397       \seq_clear:N \l_tmpa_tl
```

```

398     \seq_map_inline:Nn #1 {
399         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
400         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
401     }
402     \seq_set_eq:NN #1 \l_tmpa_tl
403 }
404 \stex_path_canonicalize:N #1
405 }
406 }
407

```

(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`

```

408 \cs_new_protected:Nn \stex_path_to_string:NN {
409     \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
410 }
411
412 \cs_new:Nn \stex_path_to_string:N {
413     \seq_use:Nn #1 /
414 }

```

(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`

```

415 \str_const:Nn \c__stex_path_dot_str {.}
416 \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.

```

417 \cs_new_protected:Nn \stex_path_canonicalize:N {
418     \seq_if_empty:NF #1 {
419         \seq_clear:N \l_tmpa_seq
420         \seq_get_left:NN #1 \l_tmpa_tl
421         \str_if_empty:NT \l_tmpa_tl {
422             \seq_put_right:Nn \l_tmpa_seq {}
423         }
424         \seq_map_inline:Nn #1 {
425             \str_set:Nn \l_tmpa_tl { ##1 }
426             \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
427                 \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
428                     \seq_if_empty:NNTF \l_tmpa_seq {
429                         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
430                             \c__stex_path_up_str
431                         }
432                     }{
433                         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
434                         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
435                             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
436                                 \c__stex_path_up_str
437                             }
438                         }{

```

```

439         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
440     }
441 }
442 }{
443     \str_if_empty:NF \l_tmpa_tl {
444         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
445     }
446 }
447 }
448 }
449 \seq_gset_eq:NN #1 \l_tmpa_seq
450 }
451 }

```

(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}`

```

452 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
453     \seq_if_empty:NTF #1 {
454         \prg_return_false:
455     }{
456         \seq_get_left:NN #1 \l_tmpa_tl
457         \sys_if_platform_windows:TF{
458             \str_if_in:NnTF \l_tmpa_tl {:}{
459                 \prg_return_true:
460             }{
461                 \prg_return_false:
462             }
463         }{
464             \str_if_empty:NTF \l_tmpa_tl {
465                 \prg_return_true:
466             }{
467                 \prg_return_false:
468             }
469         }
470     }
471 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

472 \str_new:N\l_stex_kpsewhich_return_str
473 \cs_new_protected:Nn \stex_kpsewhich:n {
474     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
475     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
476     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
477 }

```

(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`

```

478 \sys_if_platform_windows:TF{
479   \begingroup\escapechar=-1\catcode'\=12
480   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
481   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
482   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
483   }}{
484   \stex_kpsewhich:n{-var-value~PWD}
485   }
486
487 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
488 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
489 \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

490 `<@@=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

```

491 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

492 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
493 \stex_path_from_string:Nn \c_stex_mainfile_seq
494   \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`

```

495 \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`

```

496 \cs_new_protected:Nn \stex_filestack_push:n {
497   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
498   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
499     \stex_path_from_string:Nn\g_stex_currentfile_seq{
500       \c_stex_pwd_str/#1
501     }
502   }
503   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
504   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
505 }

```

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

`\stex_filestack_pop:`

```

506 \cs_new_protected:Nn \stex_filestack_pop: {
507   \seq_if_empty:NF\g__stex_files_stack{
508     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
509   }
510   \seq_if_empty:NTF\g__stex_files_stack{
511     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
512   }{
513     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
514     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
515   }
516 }
```

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

Hooks for the current file:

```

517 \AddToHook{file/before}{
518   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
519 }
520 \AddToHook{file/after}{
521   \stex_filestack_pop:
522 }
```

25.4 MathHub Repositories

523 `<@@=stex_mathhub>`

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

`\c_stex_mathhub_str`

```

524 \str_if_empty:NTF\mathhub{
525   \sys_if_platform_windows:TF{
526     \begingroup\escapechar=-1\catcode'\=12
527     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
528     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
529     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
530   }{
531     \stex_kpsewhich:n{-var-value-MATHHUB}
532   }
533   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
534 }
535 \str_if_empty:NT \c_stex_mathhub_str {
536   \sys_if_platform_windows:TF{
537     \begingroup\escapechar=-1\catcode'\=12
538     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
539     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
540     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
541   }{
542     \stex_kpsewhich:n{-var-value-HOME}
543   }
544   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
545     \begingroup\escapechar=-1\catcode'\=12
546     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```

547     \sys_if_platform_windows:T{
548       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
549     }
550     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
551     \endgroup
552     \ior_close:N \l_tmpa_ior
553   }
554 }
555 \str_if_empty:NTF\c_stex_mathhub_str{
556   \msg_warning:nn{stex}{warning/nomathhub}
557 }{
558   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
559   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
560 }
561 }{
562   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
563   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
564     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
565       \c_stex_pwd_str/\mathhub
566     }
567   }
568   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
569   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
570 }

```

(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

```

571 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
572   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
573     \str_set:Nx \l_tmpa_str { #1 }
574     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
575     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
576     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
577     \_stex_mathhub_find_manifest:N \l_tmpa_seq
578     \seq_if_empty:NTF \_stex_mathhub_manifest_file_seq {
579       \msg_error:nnxx{stex}{error/norepository}{#1}{
580         \stex_path_to_string:N \c_stex_mathhub_str
581       }
582     } {
583       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
584     }
585   }
586 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```

587 \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`:

```

588 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
589   \seq_set_eq:NN \l_tmpa_seq #1
590   \bool_set_true:N \l_tmpa_bool
591   \bool_while_do:Nn \l_tmpa_bool {
592     \seq_if_empty:NTF \l_tmpa_seq {
593       \bool_set_false:N \l_tmpa_bool
594     }{
595       \file_if_exist:nTF{
596         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
597       }{
598         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
599         \bool_set_false:N \l_tmpa_bool
600       }{
601         \file_if_exist:nTF{
602           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
603         }{
604           \seq_put_right:Nn \l_tmpa_seq{META-INF}
605           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
606           \bool_set_false:N \l_tmpa_bool
607         }{
608           \file_if_exist:nTF{
609             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
610           }{
611             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
612             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
613             \bool_set_false:N \l_tmpa_bool
614           }{
615             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
616           }
617         }
618       }
619     }
620   }
621   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
622 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

623 \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:

```

624 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
625   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
626   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
627   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
628     \str_set:Nn \l_tmpa_str {##1}
629     \exp_args:NNoo \seq_set_split:Nnn
630       \l_tmpb_seq \c_colon_str \l_tmpa_str
631     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

632 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
633 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
634 }
635 \exp_args:No \str_case:nnTF \l_tmpa_tl {
636 {id} {
637 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
638 { id } \l_tmpb_tl
639 }
640 {narration-base} {
641 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
642 { narr } \l_tmpb_tl
643 }
644 {url-base} {
645 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
646 { docurl } \l_tmpb_tl
647 }
648 {source-base} {
649 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
650 { ns } \l_tmpb_tl
651 }
652 {ns} {
653 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
654 { ns } \l_tmpb_tl
655 }
656 {dependencies} {
657 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
658 { deps } \l_tmpb_tl
659 }
660 }{}{}
661 }{}
662 }
663 \ior_close:N \c__stex_mathhub_manifest_ior
664 \stex_persist:x {
665 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
666 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
667 }
668 }
669 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

670 \cs_new_protected:Nn \stex_set_current_repository:n {
671 \stex_require_repository:n { #1 }
672 \prop_set_eq:Nc \l_stex_current_repository_prop {
673 c_stex_mathhub_#1_manifest_prop
674 }
675 }

```

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

`\stex_require_repository:n`

```

676 \cs_new_protected:Nn \stex_require_repository:n {
677 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
678 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

679   \_stex_mathhub_do_manifest:n { #1 }
680   }
681 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

682 %\prop_new:N \l_stex_current_repository_prop
683 \bool_if:NF \c_stex_persist_mode_bool {
684   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
685   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
686     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
687   } {
688     \_stex_mathhub_parse_manifest:n { main }
689     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
690     \l_tmpa_str
691     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
692     \c_stex_mathhub_main_manifest_prop
693     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
694     \stex_debug:nn{mathhub}{Current~repository:~
695     \prop_item:Nn \l_stex_current_repository_prop {id}
696   }
697 }
698 }

```

(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.

```

699 \cs_new_protected:Nn \stex_in_repository:nn {
700   \str_set:Nx \l_tmpa_str { #1 }
701   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
702   \str_if_empty:NTF \l_tmpa_str {
703     \prop_if_exist:NTF \l_stex_current_repository_prop {
704       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
705       \exp_args:Ne \l_tmpa_cs{
706         \prop_item:Nn \l_stex_current_repository_prop { id }
707       }
708     }{
709       \l_tmpa_cs{}
710     }
711   }{
712     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
713     \stex_require_repository:n \l_tmpa_str
714     \str_set:Nx \l_tmpa_str { #1 }
715     \exp_args:Nne \use:nn {
716       \stex_set_current_repository:n \l_tmpa_str
717       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
718     }{
719       \stex_debug:nn{mathhub}{switching~back~to:~
720       \prop_if_exist:NTF \l_stex_current_repository_prop {
721         \prop_item:Nn \l_stex_current_repository_prop { id }::~
722       \meaning\l_stex_current_repository_prop
723     }{

```

```

724         no~repository
725     }
726 }
727 \prop_if_exist:NTF \l_stex_current_repository_prop {
728     \stex_set_current_repository:n {
729         \prop_item:Nn \l_stex_current_repository_prop { id }
730     }
731 }{
732     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
733 }
734 }
735 }
736 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

737 \def \mhpath #1 #2 {
738     \exp_args:Ne \tl_if_empty:nTF{#1}{
739         \c_stex_mathhub_str /
740         \prop_item:Nn \l_stex_current_repository_prop { id }
741         / source / #2
742     }{
743         \c_stex_mathhub_str / #1 / source / #2
744     }
745 }

```

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

`\inputref`

`\mhinput`

```

746 \newif \ifinputref \inputreffalse
747
748 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
749     \stex_in_repository:nn {#1} {
750         \ifinputref
751             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
752         \else
753             \inputreftrue
754             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
755         \inputreffalse
756     \fi
757 }
758 }
759 \NewDocumentCommand \mhinput { 0{} m }{
760     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
761 }
762
763 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
764     \stex_in_repository:nn {#1} {
765         \stex_html_backend:TF {
766             \str_clear:N \l_tmpa_str

```

```

767     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
768       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
769     }
770     \stex_annotate_invisible:nnn{inputref}{
771       \l_tmpa_str / #2
772     }{}
773   }{
774     \begingroup
775     \inputreftrue
776     \tl_if_empty:nTF{ ##1 }{
777       \input{#2}
778     }{
779       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
780     }
781     \endgroup
782   }
783 }
784 }
785 \NewDocumentCommand \inputref { 0{ } m }{
786   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
787 }

```

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

`\addmhbibresource`

```

788 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
789   \stex_in_repository:nn {#1} {
790     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
791   }
792 }
793 \newcommand\addmhbibresource[2][]{
794   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
795 }

```

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

`\libinput`

```

796 \cs_new_protected:Npn \libinput #1 {
797   \prop_if_exist:NF \l_stex_current_repository_prop {
798     \msg_error:nnn{stex}{error/notinarchive}\libinput
799   }
800   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
801     \msg_error:nnn{stex}{error/notinarchive}\libinput
802   }
803   \seq_clear:N \l__stex_mathhub_libinput_files_seq
804   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
805   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
806
807   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
808     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
809     \IfFileExists{ \l_tmpa_str }{
810       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
811     }{}
812     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
813     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

814 }
815
816 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
817 \IfFileExists{ \l_tmpa_str }{
818   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
819 }{}
820
821 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
822   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
823 }{
824   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
825     \input{ ##1 }
826   }
827 }
828 }

```

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

`\libusepackage`

```

829 \NewDocumentCommand \libusepackage {0{ } m} {
830   \prop_if_exist:NF \l_stex_current_repository_prop {
831     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
832   }
833   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
834     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
835   }
836   \seq_clear:N \l__stex_mathhub_libinput_files_seq
837   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
838   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
839
840   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
841     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
842     \IfFileExists{ \l_tmpa_str.sty }{
843       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
844     }{}
845     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
846     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
847   }
848
849   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
850   \IfFileExists{ \l_tmpa_str.sty }{
851     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
852   }{}
853
854   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
855     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
856   }{
857     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
858       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
859         \usepackage[#1]{ ##1 }
860       }
861     }{
862       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
863     }
864   }
865 }

```

```

864 }
865 }

```

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

```

\mhgraphics
\cmhgraphics

```

```

866
867 \AddToHook{begindocument}{
868 \ltx@ifpackageloaded{graphicx}{
869   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
870   \newcommand\mhgraphics[2][]{\%
871     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
872     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
873   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
874 }{}

```

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

```

\lstinputmhlisting
\clstinputmhlisting

```

```

875 \ltx@ifpackageloaded{listings}{
876   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
877   \newcommand\lstinputmhlisting[2][]{\%
878     \def\lst@mhrepos{}\setkeys{lst}{#1}%
879     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
880   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
881 }{}
882 }
883
884 \end{package}

```

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

Chapter 26

STEX -References Implementation

```
885 <*package>
886
887 %%%%%%%%%% references.dtx %%%%%%%%%%
888
889 <@@=stex_refs>
      Warnings and error messages
890
```

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

```
891 %\iow_new:N \c__stex_refs_refs_iow
892 \AtBeginDocument{
893 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
894 }
895 \AtEndDocument{
896 % \iow_close:N \c__stex_refs_refs_iow
897 }
```

`\STEXreftitle`

```
898 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
899
900 \NewDocumentCommand \STEXreftitle { m } {
901   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
902 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
903 \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:`

```
904 \cs_new_protected:Nn \stex_get_document_uri: {  
905   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
906   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
907   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
908   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
909   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
910  
911   \str_clear:N \l_tmpa_str  
912   \prop_if_exist:NT \l_stex_current_repository_prop {  
913     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
914       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
915     }  
916   }  
917  
918   \str_if_empty:NTF \l_tmpa_str {  
919     \str_set:Nx \l_stex_current_docns_str {  
920       file:/\stex_path_to_string:N \l_tmpa_seq  
921     }  
922   }{  
923     \bool_set_true:N \l_tmpa_bool  
924     \bool_while_do:Nn \l_tmpa_bool {  
925       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
926       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
927         {source} { \bool_set_false:N \l_tmpa_bool }  
928       }{}{  
929         \seq_if_empty:NT \l_tmpa_seq {  
930           \bool_set_false:N \l_tmpa_bool  
931         }  
932       }  
933     }  
934  
935     \seq_if_empty:NTF \l_tmpa_seq {  
936       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
937     }{  
938       \str_set:Nx \l_stex_current_docns_str {  
939         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
940       }  
941     }  
942   }  
943 }
```

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

`\l_stex_current_docurl_str`

```
944 \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:`

```
945 \cs_new_protected:Nn \stex_get_document_url: {  
946   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
947   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
948   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

949 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
950 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
951
952 \str_clear:N \l_tmpa_str
953 \prop_if_exist:NT \l_stex_current_repository_prop {
954   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
955     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
956       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
957     }
958   }
959 }
960
961 \str_if_empty:NTF \l_tmpa_str {
962   \str_set:Nx \l_stex_current_docurl_str {
963     file:/\stex_path_to_string:N \l_tmpa_seq
964   }
965 }{
966   \bool_set_true:N \l_tmpa_bool
967   \bool_while_do:Nn \l_tmpa_bool {
968     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
969     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
970       {source} { \bool_set_false:N \l_tmpa_bool }
971     }{}{
972       \seq_if_empty:NT \l_tmpa_seq {
973         \bool_set_false:N \l_tmpa_bool
974       }
975     }
976   }
977
978   \seq_if_empty:NTF \l_tmpa_seq {
979     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
980   }{
981     \str_set:Nx \l_stex_current_docurl_str {
982       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
983     }
984   }
985 }
986 }

```

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

26.2 Setting Reference Targets

```

987 \str_const:Nn \c__stex_refs_url_str{URL}
988 \str_const:Nn \c__stex_refs_ref_str{REF}
989 \str_new:N \l__stex_refs_curr_label_str
990 % @currentlabel -> number
991 % @currentlabelname -> title
992 % @currentHref -> name.number <- id of some kind
993 % \theH# -> \arabic{section}
994 % \the# -> number
995 % \hyper@makecurrent{#}
996 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

997 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
998   \stex_get_document_uri:
999   \str_clear:N \l__stex_refs_curr_label_str
1000   \str_set:Nx \l_tmpa_str { #1 }
1001   \str_if_empty:NT \l_tmpa_str {
1002     \int_incr:N \l__stex_refs_unnamed_counter_int
1003     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1004   }
1005   \str_set:Nx \l__stex_refs_curr_label_str {
1006     \l_stex_current_docns_str?\l_tmpa_str
1007   }
1008   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1009     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1010   }
1011   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1012     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1013   }
1014   \stex_if_smsmode:TF {
1015     \stex_get_document_url:
1016     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1017     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1018   }{
1019     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1020     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1021     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1022     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1023   }
1024 }

```

(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.

```

1025 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1026   \str_set:Nn \l_tmpa_str {#1?#2}
1027   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1028   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1029     \seq_new:c {g__stex_refs_labels_#2_seq}
1030   }
1031   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1032     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1033   }
1034 }

```

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

```

1035 \AtEndDocument{
1036   \def\stexauxadddocref#1 #2 {}{}
1037 }

```

`\stex_ref_new_sym_target:n`

```

1038 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1039   \stex_if_smsmode:TF {
1040     \str_if_exist:cF{sref_sym_#1_type}{
1041       \stex_get_document_url:
1042       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1043     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1044   }
1045   ){
1046     \str_if_empty:NF \l__stex_refs_curr_label_str {
1047       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1048       \immediate\write\@auxout{
1049         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1050           \l__stex_refs_curr_label_str
1051         }
1052       }
1053     }
1054   }
1055 }

```

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

26.3 Using References

```

1056 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1057
1058 \keys_define:nn { stex / sref } {
1059   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1060   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1061   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1062   post          .tl_set:N = \l__stex_refs_post_tl ,
1063 }
1064 \cs_new_protected:Nn \__stex_refs_args:n {
1065   \tl_clear:N \l__stex_refs_linktext_tl
1066   \tl_clear:N \l__stex_refs_fallback_tl
1067   \tl_clear:N \l__stex_refs_pre_tl
1068   \tl_clear:N \l__stex_refs_post_tl
1069   \str_clear:N \l__stex_refs_repo_str
1070   \keys_set:nn { stex / sref } { #1 }
1071 }

```

The actual macro:

```

1072 \NewDocumentCommand \sref { 0{} m}{
1073   \__stex_refs_args:n { #1 }
1074   \str_if_empty:NTF \l__stex_refs_indocument_str {
1075     \str_set:Nx \l_tmpa_str { #2 }
1076     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1077     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1078       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1079         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1080           \str_clear:N \l_tmpa_str
1081         }
1082       }{
1083         \str_clear:N \l_tmpa_str
1084       }
1085     }{
1086       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1087       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1088 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1089 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1090   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1091   \str_clear:N \l_tmpa_str
1092   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1093     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1094       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1095     }{
1096       \seq_map_break:n {
1097         \str_set:Nn \l_tmpa_str { ##1 }
1098       }
1099     }
1100   }
1101 }{
1102   \str_clear:N \l_tmpa_str
1103 }
1104 }
1105 \str_if_empty:NTF \l_tmpa_str {
1106   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1107 }{
1108   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1109     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1110       \cs_if_exist:cTF{autoref}{
1111         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1112       }{
1113         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1114       }
1115     }{
1116       \ltx@ifpackageloaded{hyperref}{
1117         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1118       }{
1119         \l__stex_refs_linktext_tl
1120       }
1121     }
1122   }{
1123     \ltx@ifpackageloaded{hyperref}{
1124       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1125     }{
1126       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1127     }
1128   }
1129 }
1130 }{
1131   % TODO
1132 }
1133 }

```

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

`\srefsym`

```

1134 \NewDocumentCommand \srefsym { 0{} m }{
1135   \stex_get_symbol:n { #2 }
1136   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1137 }

```

```

1138
1139 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1140   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1141     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1142   }{
1143     \__stex_refs_args:n { #1 }
1144     \str_if_empty:NTF \l__stex_refs_indocument_str {
1145       \tl_if_exist:cTF{sref_sym_#2 _type}{
1146         % doc uri in \l_tmpb_str
1147         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1148         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1149           % reference
1150           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1151             \cs_if_exist:cTF{autoref}{
1152               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1153             }{
1154               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1155             }
1156           }{
1157             \ltx@ifpackageloaded{hyperref}{
1158               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1159             }{
1160               \l__stex_refs_linktext_tl
1161             }
1162           }
1163         }{
1164           % URL
1165           \ltx@ifpackageloaded{hyperref}{
1166             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1167           }{
1168             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1169           }
1170         }
1171       }{
1172         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1173       }
1174     }{
1175       % TODO
1176     }
1177   }
1178 }

```

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

\srefsymuri

```

1179 \cs_new_protected:Npn \srefsymuri #1 #2 {
1180   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1181 }

```

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

```

1182 </package>

```

Chapter 27

STEX -Modules Implementation

```
1183 <*package>
1184
1185 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1186
1187 <@@=stex_modules>
1188
1189     Warnings and error messages
1190 \msg_new:nnn{stex}{error/unknownmodule}{
1191     No~module~#1~found
1192 }
1193 \msg_new:nnn{stex}{error/syntax}{
1194     Syntax~error:~#1
1195 }
1196 \msg_new:nnn{stex}{error/siglanguage}{
1197     Module~#1~declares~signature~#2,~but~does~not~
1198     declare~its~language
1199 }
1200 \msg_new:nnn{stex}{warning/deprecated}{
1201     #1~is~deprecated;~please~use~#2~instead!
1202 }
1203 \msg_new:nnn{stex}{error/conflictingmodules}{
1204     Conflicting~imports~for~module~#1
1205 }
1206
1207 \l_stex_current_module_str The current module:
1208 \str_new:N \l_stex_current_module_str
1209
1210 (End definition for \l_stex_current_module_str. This variable is documented on page 71.)
1211
1212 \l_stex_all_modules_seq Stores all available modules
1213 \seq_new:N \l_stex_all_modules_seq
1214
1215 (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
1207 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1208   \str_if_empty:NTF \l_stex_current_module_str
1209   \prg_return_false: \prg_return_true:
1210 }

(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
1211 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1212   \prop_if_exist:cTF { c_stex_module_#1_prop }
1213   \prg_return_true: \prg_return_false:
1214 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1215 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1216   \stex_add_to_current_module:n { #1 }
1217   \stex_do_up_to_module:n { #1 }
1218 }}
1219 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1220
1221 \cs_new_protected:Nn \stex_add_to_current_module:n {
1222   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1223 }
1224 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1225 \cs_new_protected:Npn \STEXexport {
1226   \begingroup
1227   \newlinechar=-1\relax
1228   \endlinechar=-1\relax
1229   %\catcode'\ = 9\relax
1230   \expandafter\endgroup\__stex_modules_export:n
1231 }
1232 \cs_new_protected:Nn \__stex_modules_export:n {
1233   \ignorespaces #1
1234   \stex_add_to_current_module:n { \ignorespaces #1 }
1235   \stex_smsmode_do:
1236 }
1237 \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
1238 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1239   \str_set:Nx \l_tmpa_str { #1 }
1240   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1241 }

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

```


`\stex_add_import_to_current_module:n`

```

1242 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1243   \str_set:Nx \l_tmpa_str { #1 }
1244   \exp_args:Nno
1245   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1246     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1247   }
1248 }

```

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

`\stex_collect_imports:n`

```

1249 \cs_new_protected:Nn \stex_collect_imports:n {
1250   \seq_clear:N \l_stex_collect_imports_seq
1251   \__stex_modules_collect_imports:n {#1}
1252 }
1253 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1254   \seq_map_inline:cn {c_stex_module_#1_imports} {
1255     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1256       \__stex_modules_collect_imports:n { ##1 }
1257     }
1258   }
1259   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1260     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1261   }
1262 }

```

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

`\stex_do_up_to_module:n`

```

1263 \int_new:N \l__stex_modules_group_depth_int
1264 \cs_new_protected:Nn \stex_do_up_to_module:n {
1265   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1266     #1
1267   }{
1268     #1
1269     \expandafter \tl_gset:Nn
1270     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1271     \expandafter\expandafter\expandafter\endcsname
1272     \expandafter\expandafter\expandafter { \csname
1273       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1274     \aftergroup\__stex_modules_aftergroup_do:
1275   }
1276 }
1277 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1278 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1279   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1280     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1281   }}
1282   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1283     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1284     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1285   }{
1286     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1287     \aftergroup\__stex_modules_aftergroup_do:
1288   }
1289 }
1290 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1291   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1292 }

```

(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).

```

1293

```

(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.

```

1294 \str_new:N \l_stex_module_ns_str
1295 \str_new:N \l_stex_module_subpath_str
1296 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1297   \seq_set_eq:NN \l_tmpa_seq #2
1298   % split off file extension
1299   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1300   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1301   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1302   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1303
1304   \bool_set_true:N \l_tmpa_bool
1305   \bool_while_do:Nn \l_tmpa_bool {
1306     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1307     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1308       {source} { \bool_set_false:N \l_tmpa_bool }
1309     }{}{
1310       \seq_if_empty:NT \l_tmpa_seq {
1311         \bool_set_false:N \l_tmpa_bool
1312       }
1313     }
1314   }
1315
1316   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1317   % \l_tmpa_seq <- sub-path relative to archive
1318   \str_if_empty:NTF \l_stex_module_subpath_str {
1319     \str_set:Nx \l_stex_module_ns_str {#1}
1320   }{
1321     \str_set:Nx \l_stex_module_ns_str {
1322       #1/\l_stex_module_subpath_str
1323     }
1324   }
1325 }
1326
1327 \cs_new_protected:Nn \stex_modules_current_namespace: {
1328   \str_clear:N \l_stex_module_subpath_str
1329   \prop_if_exist:NTF \l_stex_current_repository_prop {
1330     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1331     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1332   }{
1333     % split off file extension
1334     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1335     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1336     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1337     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1338     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1339     \str_set:Nx \l_stex_module_ns_str {
1340       file:/\stex_path_to_string:N \l_tmpa_seq
1341     }
1342   }
1343 }

```

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

27.1 The smodule environment

smodule arguments:

```

1344 \keys_define:nn { stex / module } {
1345   title      .tl_set:N      = \smodulename ,
1346   type       .str_set_x:N   = \smodulename ,
1347   id         .str_set_x:N   = \smoduleid ,
1348   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1349   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1350   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1351   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1352   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1353   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1354   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1355   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1356 }
1357
1358 \cs_new_protected:Nn \__stex_modules_args:n {
1359   \str_clear:N \smodulename
1360   \str_clear:N \smodulename
1361   \str_clear:N \smoduleid
1362   \str_clear:N \l_stex_module_ns_str
1363   \str_clear:N \l_stex_module_deprecate_str
1364   \str_clear:N \l_stex_module_lang_str
1365   \str_clear:N \l_stex_module_sig_str
1366   \str_clear:N \l_stex_module_creators_str
1367   \str_clear:N \l_stex_module_contributors_str
1368   \str_clear:N \l_stex_module_meta_str
1369   \str_clear:N \l_stex_module_srccite_str
1370   \keys_set:nn { stex / module } { #1 }
1371 }
1372
1373 % module parameters here? In the body?
1374

```

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

```

1375 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1376 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1377 \str_set:Nx \l_stex_module_name_str { #2 }
1378 \__stex_modules_args:n { #1 }

```

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

```

1379 \stex_if_in_module:TF {
1380   % Nested module
1381   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1382   { ns } \l_stex_module_ns_str
1383   \str_set:Nx \l_stex_module_name_str {
1384     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1385     { name } / \l_stex_module_name_str
1386   }
1387   \str_if_empty:NT \l_stex_module_lang_str {
1388     \str_set:Nx \l_stex_module_lang_str {
1389       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1390       { lang }
1391     }
1392   }
1393 }{
1394   % not nested:
1395   \str_if_empty:NT \l_stex_module_ns_str {
1396     \stex_modules_current_namespace:
1397     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1398       / { \l_stex_module_ns_str }
1399     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1400     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1401       \str_set:Nx \l_stex_module_ns_str {
1402         \stex_path_to_string:N \l_tmpa_seq
1403       }
1404     }
1405   }
1406 }

```

Next, we determine the language of the module:

```

1407 \str_if_empty:NT \l_stex_module_lang_str {
1408   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1409   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1410   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1411   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1412     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1413       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1414     }
1415   }
1416   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1417   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1418     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1419     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1420       inferred~from~file~name}
1421   }
1422 }
1423
1424 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1425     \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1426   }}

```

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

```

1427   \str_if_empty:NTF \l_stex_module_sig_str {
1428     \exp_args:Nnx \prop_gset_from_keyval:cn {
1429       c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1430     } {
1431       name      = \l_stex_module_name_str ,
1432       ns        = \l_stex_module_ns_str ,
1433       file      = \exp_not:o { \g_stex_currentfile_seq } ,
1434       lang      = \l_stex_module_lang_str ,
1435       sig       = \l_stex_module_sig_str ,
1436       deprecate = \l_stex_module_deprecate_str ,
1437       meta      = \l_stex_module_meta_str
1438     }
1439     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1440     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1441     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1442     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1443     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1444   \str_if_empty:NT \l_stex_module_meta_str {
1445     \str_set:Nx \l_stex_module_meta_str {
1446       \c_stex_metatheory_ns_str ? Metatheory
1447     }
1448   }
1449   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1450     \bool_set_true:N \l_stex_in_meta_bool
1451     \exp_args:Nx \stex_add_to_current_module:n {
1452       \bool_set_true:N \l_stex_in_meta_bool
1453       \stex_activate_module:n {\l_stex_module_meta_str}
1454       \bool_set_false:N \l_stex_in_meta_bool
1455     }
1456     \stex_activate_module:n {\l_stex_module_meta_str}
1457     \bool_set_false:N \l_stex_in_meta_bool
1458   }
1459 }{
1460   \str_if_empty:NT \l_stex_module_lang_str {
1461     \msg_error:nnxx{stex}{error/siglanguage}{
1462       \l_stex_module_ns_str?\l_stex_module_name_str
1463     }\l_stex_module_sig_str}
1464   }
1465   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1466   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1467     \stex_debug:nn{modules}{(already exists)}
1468   }{
1469     \stex_debug:nn{modules}{(needs loading)}
1470     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1471     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1472     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1473     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1474 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1475 \str_set:Nx \l_tmpa_str {
1476   \stex_path_to_string:N \l_tmpa_seq /
1477   \l_tmpa_str . \l_stex_module_sig_str .tex
1478 }
1479 \IfFileExists \l_tmpa_str {
1480   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1481     \str_clear:N \l_stex_current_module_str
1482     \seq_clear:N \l_stex_all_modules_seq
1483     \stex_debug:nn{modules}{Loading~signature}
1484   }
1485   {{
1486     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1487   }
1488 }
1489 \stex_if_smsmode:F {
1490   \stex_activate_module:n {
1491     \l_stex_module_ns_str ? \l_stex_module_name_str
1492   }
1493 }
1494 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1495 }
1496 \str_if_empty:NF \l_stex_module_deprecate_str {
1497   \msg_warning:nnxx{stex}{warning/deprecated}{
1498     Module~\l_stex_current_module_str
1499   }{
1500     \l_stex_module_deprecate_str
1501   }
1502 }
1503 \seq_put_right:Nx \l_stex_all_modules_seq {
1504   \l_stex_module_ns_str ? \l_stex_module_name_str
1505 }
1506 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1507 }

```

(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}
1508 \cs_new_protected:Nn \__stex_modules_begin_module: {
1509   \stex_reactivate_macro:N \STEXexport
1510   \stex_reactivate_macro:N \importmodule
1511   \stex_reactivate_macro:N \symdecl
1512   \stex_reactivate_macro:N \notation
1513   \stex_reactivate_macro:N \symdef
1514
1515   \stex_debug:nn{modules}{
1516     New~module:\\
1517     Namespace:~\l_stex_module_ns_str\\
1518     Name:~\l_stex_module_name_str\\
1519     Language:~\l_stex_module_lang_str\\
1520     Signature:~\l_stex_module_sig_str\\
1521     Metatheory:~\l_stex_module_meta_str\\

```

```

1522   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1523 }
1524
1525 \stex_if_do_html:T{
1526   \begin{stex_annotate_env} {theory} {
1527     \l_stex_module_ns_str ? \l_stex_module_name_str
1528   }
1529
1530   \stex_annotate_invisible:nnn{header}{} {
1531     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1532     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1533     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1534       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1535     }
1536     \str_if_empty:NF \smoduletype {
1537       \stex_annotate:nnn{type}{\smoduletype}{}
1538     }
1539   }
1540 }
1541 % TODO: Inherit metatheory for nested modules?
1542 }
1543 \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}

```

1544 \cs_new_protected:Nn \_stex_modules_end_module: {
1545   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1546   \stex_reset_up_to_module:n \l_stex_current_module_str
1547   \stex_if_smsmode:T {
1548     \stex_persist:x {
1549       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1550         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1551       }
1552       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1553         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1554       }
1555       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1556         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1557       }
1558       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1559     }
1560     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1561     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1562   }
1563 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1564 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1565 \NewDocumentEnvironment { smodule } { 0 } { m } {
1566   \stex_module_setup:nn{#1}{#2}
1567   \par
1568   \stex_if_smsmode:F{

```

```

1569 \tl_if_empty:NF \smodulename {
1570 \exp_args:No \stex_document_title:n \smodulename
1571 }
1572 \tl_clear:N \l_tmpa_tl
1573 \clist_map_inline:Nn \smodulename {
1574 \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1575 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1576 }
1577 }
1578 \tl_if_empty:NTF \l_tmpa_tl {
1579 \__stex_modules_smodule_start:
1580 }{
1581 \l_tmpa_tl
1582 }
1583 }
1584 \__stex_modules_begin_module:
1585 \str_if_empty:NF \smoduleid {
1586 \stex_ref_new_doc_target:n \smoduleid
1587 }
1588 \stex_smsmode_do:
1589 } {
1590 \__stex_modules_end_module:
1591 \stex_if_smsmode:F {
1592 \end{stex_annotate_env}
1593 \clist_set:No \l_tmpa_clist \smodulename
1594 \tl_clear:N \l_tmpa_tl
1595 \clist_map_inline:Nn \l_tmpa_clist {
1596 \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1597 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1598 }
1599 }
1600 \tl_if_empty:NTF \l_tmpa_tl {
1601 \__stex_modules_smodule_end:
1602 }{
1603 \l_tmpa_tl
1604 }
1605 }
1606 }

```

\stexpatchmodule

```

1607 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1608 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1609
1610 \newcommand\stexpatchmodule[3] [] {
1611 \str_set:Nx \l_tmpa_str{ #1 }
1612 \str_if_empty:NTF \l_tmpa_str {
1613 \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1614 \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1615 }{
1616 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1617 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1618 }
1619 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1620 \NewDocumentCommand \STEXModule { m } {
1621   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1622   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1623   \tl_set:Nn \l_tmpa_tl {
1624     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1625   }
1626   \seq_map_inline:Nn \l_stex_all_modules_seq {
1627     \str_set:Nn \l_tmpb_str { ##1 }
1628     \str_if_eq:eeT { \l_tmpa_str } {
1629       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1630     } {
1631       \seq_map_break:n {
1632         \tl_set:Nn \l_tmpa_tl {
1633           \stex_invoke_module:n { ##1 }
1634         }
1635       }
1636     }
1637   }
1638   \l_tmpa_tl
1639 }
1640
1641 \cs_new_protected:Nn \stex_invoke_module:n {
1642   \stex_debug:nn{modules}{Invoking~module~#1}
1643   \peek_charcode_remove:NTF ! {
1644     \__stex_modules_invoke_uri:nN { #1 }
1645   } {
1646     \peek_charcode_remove:NTF ? {
1647       \__stex_modules_invoke_symbol:nn { #1 }
1648     } {
1649       \msg_error:nnx{stex}{error/syntax}{
1650         ?~or~!~expected~after~
1651         \c_backslash_str STEXModule{#1}
1652       }
1653     }
1654   }
1655 }
1656
1657 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1658   \str_set:Nn #2 { #1 }
1659 }
1660
1661 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1662   \stex_invoke_symbol:n{#1?#2}
1663 }

```

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

```

\stex_activate_module:n
1664 \bool_new:N \l_stex_in_meta_bool
1665 \bool_set_false:N \l_stex_in_meta_bool

```

```

1666 \cs_new_protected:Nn \stex_activate_module:n {
1667   \stex_debug:nn{modules}{Activating~module~#1}
1668   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1669     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1670     \use:c{ c_stex_module_#1_code }
1671   }
1672 }

```

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

```

1673 \</package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1674 <*package>
1675
1676 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1677
```

28.1 SMS Mode

```
1678 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1679 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1680 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1681 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1682
1683 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1684   \makeatletter
1685   \makeatother
1686   \ExplSyntaxOn
1687   \ExplSyntaxOff
1688   \rustexBREAK
1689 }
1690
1691 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1692   \symdef
1693   \importmodule
1694   \notation
1695   \symdecl
1696   \STEXexport
1697   \inlineass
1698   \inlinedef
1699   \inlineex
1700   \endinput
1701   \setnotation
```

```

1702 \copynotation
1703 \assign
1704 \renamedekl
1705 \donotcopy
1706 \instantiate
1707 }
1708
1709 \exp_args:Nn \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1710   \tl_to_str:n {
1711     smodule,
1712     copymodule,
1713     interpretmodule,
1714     sdefinition,
1715     sexample,
1716     sassertion,
1717     sparagraph,
1718     mathstructure
1719   }
1720 }

```

(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`

```

1721 \bool_new:N \g__stex_smsmode_bool
1722 \bool_set_false:N \g__stex_smsmode_bool
1723 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1724   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1725 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1726 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1727   \vbox_set:Nn \l_tmpa_box {
1728     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1729     \bool_gset_true:N \g__stex_smsmode_bool
1730     #2
1731     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1732   }
1733   \box_clear:N \l_tmpa_box
1734 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1735 \quark_new:N \q__stex_smsmode_break
1736
1737 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1738   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1739   \stex_smsmode_do:
1740 }
1741
1742 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1743   \_stex_modules_args:n{#1}

```

```

1744 \stex_if_in_module:F {
1745   \str_if_empty:NF \l_stex_module_sig_str {
1746     \stex_modules_current_namespace:
1747     \str_set:Nx \l_stex_module_name_str { #2 }
1748     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1749       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1750       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1751       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1752       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1753       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1754       \str_set:Nx \l_tmpa_str {
1755         \stex_path_to_string:N \l_tmpa_seq /
1756         \l_tmpa_str . \l_stex_module_sig_str .tex
1757       }
1758       \IfFileExists \l_tmpa_str {
1759         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1760       }{
1761         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1762       }
1763     }
1764   }
1765 }
1766 }
1767
1768 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1769   \stex_filestack_push:n{#1}
1770   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1771   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1772   % ----- new -----
1773   \__stex_smsmode_in_smsmode:nn{#1}{
1774     \let\importmodule\__stex_smsmode_importmodule:
1775     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1776     \let\__stex_modules_begin_module:\relax
1777     \let\__stex_modules_end_module:\relax
1778     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1779     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{module}}
1780     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1781     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1782     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1783     \everyeof{\q__stex_smsmode_break\noexpand}
1784     \expandafter\expandafter\expandafter
1785     \stex_smsmode_do:
1786     \csname @ @ input\endcsname "#1"\relax
1787
1788     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1789       \stex_filestack_push:n{##1}
1790       \expandafter\expandafter\expandafter
1791       \stex_smsmode_do:
1792       \csname @ @ input\endcsname "##1"\relax
1793       \stex_filestack_pop:
1794     }
1795   }
1796   % ----- new -----
1797   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1798 #2
1799 % ----- new -----
1800 \begingroup
1801 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1802 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1803   \stex_import_module_uri:nn ##1
1804   \stex_import_require_module:nnnn
1805   \l_stex_import_ns_str
1806   \l_stex_import_archive_str
1807   \l_stex_import_path_str
1808   \l_stex_import_name_str
1809 }
1810 \endgroup
1811 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1812 % ----- new -----
1813 \everyeof{\q__stex_smsmode_break\noexpand}
1814 \expandafter\expandafter\expandafter
1815 \stex_smsmode_do:
1816 \csname @ @ input\endcsname "#1"\relax
1817 }
1818 \stex_filestack_pop:
1819 }

```

(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:

```

1820 \cs_new_protected:Npn \stex_smsmode_do: {
1821   \stex_if_smsmode:T {
1822     \__stex_smsmode_do:w
1823   }
1824 }
1825 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1826   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1827     \expandafter\if\expandafter\relax\noexpand#1
1828     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1829     \else\expandafter\__stex_smsmode_do:w\fi
1830   }{
1831     \__stex_smsmode_do:w % #1
1832   }
1833 }
1834 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1835   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1836     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1837       #1\__stex_smsmode_do:w
1838     }{
1839       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1840         #1
1841       }{
1842         \cs_if_eq:NNTF \begin #1 {
1843           \__stex_smsmode_check_begin:n
1844         }{
1845           \cs_if_eq:NNTF \end #1 {
1846             \__stex_smsmode_check_end:n

```

```

1847         }{
1848         \__stex_smsmode_do:w
1849         }
1850     }
1851 }
1852 }
1853 }
1854 }
1855
1856 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1857 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1858 \begin{#1}
1859 }{
1860 \__stex_smsmode_do:w
1861 }
1862 }
1863 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1864 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1865 \end{#1}\__stex_smsmode_do:w
1866 }{
1867 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1868 }
1869 }

```

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

28.2 Inheritance

```

1870 <@@=stex_importmodule>

\stex_import_module_uri:nn

1871 \cs_new_protected:Nn \stex_import_module_uri:nn {
1872 \str_set:Nx \l_stex_import_archive_str { #1 }
1873 \str_set:Nn \l_stex_import_path_str { #2 }
1874
1875 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1876 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1877 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1878
1879 \stex_modules_current_namespace:
1880 \bool_lazy_all:nTF {
1881 {\str_if_empty_p:N \l_stex_import_archive_str}
1882 {\str_if_empty_p:N \l_stex_import_path_str}
1883 {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1884 }{
1885 \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1886 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1887 }{
1888 \str_if_empty:NT \l_stex_import_archive_str {
1889 \prop_if_exist:NT \l_stex_current_repository_prop {
1890 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1891 }
1892 }
1893 \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1894     \str_if_empty:NF \l_stex_import_path_str {
1895         \str_set:Nx \l_stex_import_ns_str {
1896             \l_stex_module_ns_str / \l_stex_import_path_str
1897         }
1898     }
1899 }{
1900     \stex_require_repository:n \l_stex_import_archive_str
1901     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1902     \l_stex_import_ns_str
1903     \str_if_empty:NF \l_stex_import_path_str {
1904         \str_set:Nx \l_stex_import_ns_str {
1905             \l_stex_import_ns_str / \l_stex_import_path_str
1906         }
1907     }
1908 }
1909 }
1910 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1915 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1916     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1917
1918         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1919
1920         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/{}} {#4}
1921         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1922
1923         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1924
1925         % archive
1926         \str_set:Nx \l_tmpa_str { #2 }
1927         \str_if_empty:NTF \l_tmpa_str {
1928             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1929         } {
1930             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1931             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1932             \seq_put_right:Nn \l_tmpa_seq { source }
1933         }
1934
1935         % path
1936         \str_set:Nx \l_tmpb_str { #3 }
1937         \str_if_empty:NTF \l_tmpb_str {
1938             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1939

```



```

1940 \ltx@ifpackageloaded{babel} {
1941   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1942     { \language } \l_tmpb_str {
1943       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1944     }
1945   } {
1946     \str_clear:N \l_tmpb_str
1947   }
1948
1949   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1950   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1951     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1952   }{
1953     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1954     \IfFileExists{ \l_tmpa_str.tex }{
1955       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1956     }{
1957       % try english as default
1958       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1959       \IfFileExists{ \l_tmpa_str.en.tex }{
1960         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1961       }{
1962         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1963       }
1964     }
1965   }
1966
1967 } {
1968   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1969   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1970
1971   \ltx@ifpackageloaded{babel} {
1972     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1973       { \language } \l_tmpb_str {
1974         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1975       }
1976   } {
1977     \str_clear:N \l_tmpb_str
1978   }
1979
1980   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1981
1982   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1983   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1984     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1985   }{
1986     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1987     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1988       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1989     }{
1990       % try english as default
1991       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1992       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1993         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```

```

1994     }{
1995     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1996     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1997         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1998     }{
1999         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
2000         \IfFileExists{ \l_tmpa_str.tex }{
2001             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2002         }{
2003             % try english as default
2004             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
2005             \IfFileExists{ \l_tmpa_str.en.tex }{
2006                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2007             }{
2008                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2009             }
2010         }
2011     }
2012 }
2013 }
2014 }
2015 }
2016
2017 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2018     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2019         \seq_clear:N \l_stex_all_modules_seq
2020         \str_clear:N \l_stex_current_module_str
2021         \str_set:Nx \l_tmpb_str { #2 }
2022         \str_if_empty:NF \l_tmpb_str {
2023             \stex_set_current_repository:n { #2 }
2024         }
2025         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2026     }
2027
2028     \stex_if_module_exists:nF { #1 ? #4 } {
2029         \msg_error:nnx{stex}{error/unknownmodule}{
2030             #1?#4~(in~file~\g__stex_importmodule_file_str)
2031         }
2032     }
2033 }
2034
2035 }
2036 \stex_activate_module:n { #1 ? #4 }
2037 }

```

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

`\importmodule`

```

2038 \NewDocumentCommand \importmodule { 0{} m } {
2039     \stex_import_module_uri:nn { #1 } { #2 }
2040     \stex_debug:nn{modules}{Importing~module:~
2041         \l_stex_import_ns_str ? \l_stex_import_name_str
2042     }
2043     \stex_import_require_module:nnnn

```

```

2044 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2045 { \l_stex_import_path_str } { \l_stex_import_name_str }
2046 \stex_if_smsmode:F {
2047   \stex_annotate_invisible:nnn
2048   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2049 }
2050 \exp_args:Nx \stex_add_to_current_module:n {
2051   \stex_import_require_module:nnnn
2052   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2053   { \l_stex_import_path_str } { \l_stex_import_name_str }
2054 }
2055 \exp_args:Nx \stex_add_import_to_current_module:n {
2056   \l_stex_import_ns_str ? \l_stex_import_name_str
2057 }
2058 \stex_smsmode_do:
2059 \ignorespacesandpars
2060 }
2061 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2062 \NewDocumentCommand \usemodule { 0{} m } {
2063   \stex_if_smsmode:F {
2064     \stex_import_module_uri:nn { #1 } { #2 }
2065     \stex_import_require_module:nnnn
2066     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2067     { \l_stex_import_path_str } { \l_stex_import_name_str }
2068     \stex_annotate_invisible:nnn
2069     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2070   }
2071   \stex_smsmode_do:
2072   \ignorespacesandpars
2073 }

```

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

```

2074 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2075   \tl_if_empty:nF{#2}{
2076     \clist_set:Nn \l_tmpa_clist {#2}
2077     \clist_map_inline:Nn \l_tmpa_clist {
2078       \tl_if_head_eq_charcode:nNTF {##1}[{
2079         #1 ##1
2080       }{
2081         #1{##1}
2082       }
2083     }
2084   }
2085 }
2086 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2087
2088
2089 </package>

```

Chapter 29

STEX -Symbols Implementation

```
2090 <*package>
2091
2092 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2093
2094 Warnings and error messages
2095 \msg_new:nnn{stex}{error/wrongargs}{
2096   args~value~in~symbol~declaration~for~#1~
2097   needs~to~be~i,~a,~b~or~B,~but~#2~given
2098 }
2099 \msg_new:nnn{stex}{error/unknownsymbol}{
2100   No~symbol~#1~found!
2101 }
2102 \msg_new:nnn{stex}{error/seqlength}{
2103   Expected~#1~arguments;~got~#2!
2104 }
2105 \msg_new:nnn{stex}{error/unknownnotation}{
2106   Unknown~notation~#1~for~#2!
2107 }
```

29.1 Symbol Declarations

```
2107 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2108 \cs_new_protected:Nn \stex_all_symbols:n {
2109   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2110   \seq_map_inline:Nn \l_stex_all_modules_seq {
2111     \seq_map_inline:cn{c_stex_module_##1_constants}{
2112       \__stex_symdecl_all_symbols_cs{##1?####1}
2113     }
2114   }
2115 }
```

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

\STEXsymbol

```
2116 \NewDocumentCommand \STEXsymbol { m } {  
2117   \stex_get_symbol:n { #1 }  
2118   \exp_args:No  
2119   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2120 }
```

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

symdecl arguments:

```
2121 \keys_define:nn { stex / symdecl } {  
2122   name      .str_set_x:N = \l_stex_symdecl_name_str ,  
2123   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2124   args      .str_set_x:N = \l_stex_symdecl_args_str ,  
2125   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2126   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,  
2127   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2128   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2129   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2130   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2131   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,  
2132   assoc     .choices:nn =  
2133     {bin,binl,binr,pre,conj,pwconj}  
2134     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2135 }  
2136  
2137 \bool_new:N \l_stex_symdecl_make_macro_bool  
2138  
2139 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2140   \str_clear:N \l_stex_symdecl_name_str  
2141   \str_clear:N \l_stex_symdecl_args_str  
2142   \str_clear:N \l_stex_symdecl_deprecate_str  
2143   \str_clear:N \l_stex_symdecl_reorder_str  
2144   \str_clear:N \l_stex_symdecl_assoctype_str  
2145   \bool_set_false:N \l_stex_symdecl_local_bool  
2146   \tl_clear:N \l_stex_symdecl_type_tl  
2147   \tl_clear:N \l_stex_symdecl_definiens_tl  
2148  
2149   \keys_set:nn { stex / symdecl } { #1 }  
2150 }
```

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

```
2151  
2152 \NewDocumentCommand \symdecl { s m O{} } {  
2153   \__stex_symdecl_args:n { #3 }  
2154   \IfBooleanTF #1 {  
2155     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2156   } {  
2157     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2158   }  
2159   \stex_symdecl_do:n { #2 }  
2160   \stex_smsmode_do:  
2161 }
```

```

2162
2163 \cs_new_protected:Nn \stex_symdecl_do:nn {
2164   \__stex_symdecl_args:n{#1}
2165   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2166   \stex_symdecl_do:n{#2}
2167 }
2168
2169 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2170 \cs_new_protected:Nn \stex_symdecl_do:n {
2171   \stex_if_in_module:F {
2172     % TODO throw error? some default namespace?
2173   }
2174
2175   \str_if_empty:NT \l_stex_symdecl_name_str {
2176     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2177   }
2178
2179   \prop_if_exist:cT { l_stex_symdecl_
2180     \l_stex_current_module_str ?
2181     \l_stex_symdecl_name_str
2182     _prop
2183   }{
2184     % TODO throw error (beware of circular dependencies)
2185   }
2186
2187   \prop_clear:N \l_tmpa_prop
2188   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2189   \seq_clear:N \l_tmpa_seq
2190   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2191   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2192
2193   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2194     \str_if_empty:NF \l_stex_module_deprecate_str {
2195       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2196     }
2197   }
2198   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2199
2200   \exp_args:No \stex_add_constant_to_current_module:n {
2201     \l_stex_symdecl_name_str
2202   }
2203
2204   % arity/args
2205   \int_zero:N \l_tmpb_int
2206
2207   \bool_set_true:N \l_tmpa_bool
2208   \str_map_inline:Nn \l_stex_symdecl_args_str {
2209     \token_case_meaning:NnF ##1 {
2210       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2211       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2212     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2213     {\tl_to_str:n a} {
2214         \bool_set_false:N \l_tmpa_bool
2215         \int_incr:N \l_tmpb_int
2216     }
2217     {\tl_to_str:n B} {
2218         \bool_set_false:N \l_tmpa_bool
2219         \int_incr:N \l_tmpb_int
2220     }
2221 }{
2222     \msg_error:nnxx{stex}{error/wrongargs}{
2223         \l_stex_current_module_str ?
2224         \l_stex_symdecl_name_str
2225     }{##1}
2226 }
2227 }
2228 \bool_if:NTF \l_tmpa_bool {
2229     % possibly numeric
2230     \str_if_empty:NTF \l_stex_symdecl_args_str {
2231         \prop_put:Nnn \l_tmpa_prop { args } {}
2232         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2233     }{
2234         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2235         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2236         \str_clear:N \l_tmpa_str
2237         \int_step_inline:nn \l_tmpa_int {
2238             \str_put_right:Nn \l_tmpa_str i
2239         }
2240         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2241     }
2242 } {
2243     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2244     \prop_put:Nnx \l_tmpa_prop { arity }
2245     { \str_count:N \l_stex_symdecl_args_str }
2246 }
2247 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2248
2249 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2250     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2251 }{
2252     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2253 }
2254
2255 % semantic macro
2256
2257 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2258     \exp_args:Nx \stex_do_up_to_module:n {
2259         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2260             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2261         }}
2262     }
2263 }
2264
2265 \stex_debug:nn{symbols}{New~symbol:~

```

```

2266 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2267 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2268 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2269 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2270 }
2271
2272 % circular dependencies require this:
2273 \stex_if_do_html:T {
2274   \stex_annotate_invisible:nnn {symdecl} {
2275     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2276   } {
2277     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2278       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
2279     }
2280     \stex_annotate_invisible:nnn{args}{}{
2281       \prop_item:Nn \l_tmpa_prop { args }
2282     }
2283     \stex_annotate_invisible:nnn{macroname}{#1}{}
2284     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2285       \stex_annotate_invisible:nnn{definiens}{}
2286       {\l_stex_symdecl_definiens_tl$}
2287     }
2288     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2289       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2290     }
2291     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2292       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{}
2293     }
2294   }
2295 }
2296 \prop_if_exist:cF {
2297   l_stex_symdecl_
2298   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2299   _prop
2300 } {
2301   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2302     \__stex_symdecl_restore_symbol:nnnnnnn
2303       {\l_stex_symdecl_name_str}
2304       { \prop_item:Nn \l_tmpa_prop {args} }
2305       { \prop_item:Nn \l_tmpa_prop {arity} }
2306       { \prop_item:Nn \l_tmpa_prop {assocs} }
2307       { \prop_item:Nn \l_tmpa_prop {defined} }
2308       {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2309       {\l_stex_current_module_str}
2310   }
2311 }
2312 }
2313 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2314   \prop_clear:N \l_tmpa_prop
2315   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2316   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2317   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2318   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2319   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }

```



```

2320 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2321 \tl_if_empty:nF{#6}{
2322   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2323 }
2324 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2325 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2326 }

```

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

`\stex_get_symbol:n`

```

2327 \str_new:N \l_stex_get_symbol_uri_str
2328
2329 \cs_new_protected:Nn \stex_get_symbol:n {
2330   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2331     \tl_set:Nn \l_tmpa_tl { #1 }
2332     \__stex_symdecl_get_symbol_from_cs:
2333   }{
2334     % argument is a string
2335     % is it a command name?
2336     \cs_if_exist:cTF { #1 }{
2337       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2338       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2339       \str_if_empty:NTF \l_tmpa_str {
2340         \exp_args:Nx \cs_if_eq:NNTF {
2341           \tl_head:N \l_tmpa_tl
2342         } \stex_invoke_symbol:n {
2343           \__stex_symdecl_get_symbol_from_cs:
2344         }{
2345           \__stex_symdecl_get_symbol_from_string:n { #1 }
2346         }
2347       } {
2348         \__stex_symdecl_get_symbol_from_string:n { #1 }
2349       }
2350     }{
2351       % argument is not a command name
2352       \__stex_symdecl_get_symbol_from_string:n { #1 }
2353       % \l_stex_all_symbols_seq
2354     }
2355   }
2356   \str_if_eq:eeF {
2357     \prop_item:cn {
2358       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2359     }{ deprecate }
2360   }{
2361     \msg_warning:nxxx{stex}{warning/deprecated}{
2362       Symbol~\l_stex_get_symbol_uri_str
2363     }{
2364       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2365     }
2366   }
2367 }
2368
2369 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2370 \tl_set:Nn \l_tmpa_tl {
2371   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2372 }
2373 \str_set:Nn \l_tmpa_str { #1 }
2374
2375 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2376
2377 \str_if_in:NnTF \l_tmpa_str ? {
2378   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2379   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2380   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2381 }{
2382   \str_clear:N \l_tmpb_str
2383 }
2384 \str_if_empty:NNTF \l_tmpb_str {
2385   \seq_map_inline:Nn \l_stex_all_modules_seq {
2386     \seq_map_inline:cn{c_stex_module_###1_constants}{
2387       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2388         \seq_map_break:n{\seq_map_break:n{
2389           \tl_set:Nn \l_tmpa_tl {
2390             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2391           }
2392         }}
2393       }
2394     }
2395   }
2396 }{
2397   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2398   \seq_map_inline:Nn \l_stex_all_modules_seq {
2399     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2400       \seq_map_inline:cn{c_stex_module_###1_constants}{
2401         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2402           \seq_map_break:n{\seq_map_break:n{
2403             \tl_set:Nn \l_tmpa_tl {
2404               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2405             }
2406           }}
2407         }
2408       }
2409     }
2410   }
2411 }
2412
2413 \l_tmpa_tl
2414 }
2415
2416 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2417   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2418   { \tl_tail:N \l_tmpa_tl }
2419   \tl_if_single:NNTF \l_tmpa_tl {
2420     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2421       \exp_after:wN \str_set:Nn \exp_after:wN
2422       \l_stex_get_symbol_uri_str \l_tmpa_tl
2423     }{

```

```

2424 % TODO
2425 % tail is not a single group
2426 }
2427 }{
2428 % TODO
2429 % tail is not a single group
2430 }
2431 }

```

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

29.2 Notations

```

2432 <@@=stex_notation>

notation arguments:
2433 \keys_define:nn { stex / notation } {
2434 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2435 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2436 prec .str_set_x:N = \l__stex_notation_prec_str ,
2437 op .tl_set:N = \l__stex_notation_op_tl ,
2438 primary .bool_set:N = \l__stex_notation_primary_bool ,
2439 primary .default:n = {true} ,
2440 unknown .code:n = \str_set:Nx
2441 \l__stex_notation_variant_str \l_keys_key_str
2442 }
2443
2444 \cs_new_protected:Nn \stex_notation_args:n {
2445 % \str_clear:N \l__stex_notation_lang_str
2446 \str_clear:N \l__stex_notation_variant_str
2447 \str_clear:N \l__stex_notation_prec_str
2448 \tl_clear:N \l__stex_notation_op_tl
2449 \bool_set_false:N \l__stex_notation_primary_bool
2450
2451 \keys_set:nn { stex / notation } { #1 }
2452 }

```

\notation

```

2453 \NewDocumentCommand \notation { s m 0{}} {
2454 \stex_notation_args:n { #3 }
2455 \tl_clear:N \l_stex_symdecl_definiens_tl
2456 \stex_get_symbol:n { #2 }
2457 \tl_set:Nn \l_stex_notation_after_do_tl {
2458 \__stex_notation_final:
2459 \IfBooleanTF#1{
2460 \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2461 }{}
2462 \stex_smsmode_do:\ignorespacesandpars
2463 }
2464 \stex_notation_do:nnnnn
2465 { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2466 { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2467 { \l__stex_notation_variant_str }
2468 { \l__stex_notation_prec_str }

```

```

2469 }
2470 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2471 \seq_new:N \l__stex_notation_precedences_seq
2472 \tl_new:N \l__stex_notation_opprec_tl
2473 \int_new:N \l__stex_notation_currarg_int
2474 \tl_new:N \stex_symbol_after_invokation_tl
2475
2476 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2477   \let\l_stex_current_symbol_str\relax
2478   \seq_clear:N \l__stex_notation_precedences_seq
2479   \tl_clear:N \l__stex_notation_opprec_tl
2480   \str_set:Nx \l__stex_notation_args_str { #1 }
2481   \str_set:Nx \l__stex_notation_arity_str { #2 }
2482   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2483   \str_set:Nx \l__stex_notation_prec_str { #4 }
2484
2485   % precedences
2486   \str_if_empty:NTF \l__stex_notation_prec_str {
2487     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2488       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2489     }{
2490       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2491     }
2492   } {
2493     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2494       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2495       \int_step_inline:nn { \l__stex_notation_arity_str } {
2496         \exp_args:NNo
2497         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2498       }
2499     }{
2500       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2501       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2502         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2503         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2504           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2505             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2506           \seq_map_inline:Nn \l_tmpa_seq {
2507             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2508           }
2509         }
2510       }{
2511         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2512           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2513         }{
2514           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2515         }
2516       }
2517     }
2518   }

```

```

2519 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2520 \int_step_inline:nn { \l__stex_notation_arity_str } {
2521   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2522     \exp_args:NNo
2523     \seq_put_right:No \l__stex_notation_precedences_seq {
2524       \l__stex_notation_opprec_tl
2525     }
2526   }
2527 }
2528 }
2529 \tl_clear:N \l_stex_notation_dummyargs_tl
2530
2531 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2532   \exp_args:NNe
2533   \cs_set:Npn \l_stex_notation_macrocode_cs {
2534     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2535     { \l__stex_notation_suffix_str }
2536     { \l__stex_notation_opprec_tl }
2537     { \exp_not:n { #5 } }
2538   }
2539   \l_stex_notation_after_do_tl
2540 }{
2541   \str_if_in:NnTF \l__stex_notation_args_str b {
2542     \exp_args:Nne \use:nn
2543     {
2544       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2545       \cs_set:Npn \l__stex_notation_arity_str } { {
2546         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2547         { \l__stex_notation_suffix_str }
2548         { \l__stex_notation_opprec_tl }
2549         { \exp_not:n { #5 } }
2550       } }
2551   }{
2552     \str_if_in:NnTF \l__stex_notation_args_str B {
2553       \exp_args:Nne \use:nn
2554       {
2555         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2556         \cs_set:Npn \l__stex_notation_arity_str } { {
2557           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2558           { \l__stex_notation_suffix_str }
2559           { \l__stex_notation_opprec_tl }
2560           { \exp_not:n { #5 } }
2561         } }
2562     }{
2563       \exp_args:Nne \use:nn
2564       {
2565         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2566         \cs_set:Npn \l__stex_notation_arity_str } { {
2567           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2568           { \l__stex_notation_suffix_str }
2569           { \l__stex_notation_opprec_tl }
2570           { \exp_not:n { #5 } }
2571         } }
2572     }

```

```

2573     }
2574
2575     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2576     \int_zero:N \l__stex_notation_currarg_int
2577     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2578     \__stex_notation_arguments:
2579   }
2580 }

```

(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

```

2581 \cs_new_protected:Nn \__stex_notation_arguments: {
2582   \int_incr:N \l__stex_notation_currarg_int
2583   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2584     \l_stex_notation_after_do_tl
2585   }{
2586     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2587     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2588     \str_if_eq:VnTF \l_tmpa_str a {
2589       \__stex_notation_argument_assoc:nn{a}
2590     }{
2591       \str_if_eq:VnTF \l_tmpa_str B {
2592         \__stex_notation_argument_assoc:nn{B}
2593       }{
2594         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2595         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2596           { \stex_term_math_arg:nnn
2597             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2598             { \l_tmpb_str }
2599             { ###\int_use:N \l__stex_notation_currarg_int }
2600           }
2601         }
2602         \__stex_notation_arguments:
2603       }
2604     }
2605   }
2606 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2607 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2608
2609   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2610     {\l__stex_notation_arity_str}{
2611       #2
2612     }
2613   \int_zero:N \l_tmpa_int
2614   \tl_clear:N \l_tmpa_tl
2615   \str_map_inline:Nn \l__stex_notation_args_str {
2616     \int_incr:N \l_tmpa_int
2617     \tl_put_right:Nx \l_tmpa_tl {
2618       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```

```

2619         \str_if_eq:nnTF {##1}{B}{ {} }{
2620             {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2621             }
2622         }
2623     }
2624 }
2625 \exp_after:wN\exp_after:wN\exp_after:wN \def
2626 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2627 \exp_after:wN\exp_after:wN\exp_after:wN ##
2628 \exp_after:wN\exp_after:wN\exp_after:wN 1
2629 \exp_after:wN\exp_after:wN\exp_after:wN ##
2630 \exp_after:wN\exp_after:wN\exp_after:wN 2
2631 \exp_after:wN\exp_after:wN\exp_after:wN {
2632     \exp_after:wN \exp_after:wN \exp_after:wN
2633     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2634         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2635     }
2636 }
2637
2638 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2639 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2640     \_stex_term_math_assoc_arg:nnnn
2641     { #1\int_use:N \l__stex_notation_currarg_int }
2642     { \l_tmpa_str }
2643     { #####\int_use:N \l__stex_notation_currarg_int }
2644     { \l_tmpa_cs {####1} {####2} }
2645 } }
2646 \__stex_notation_arguments:
2647 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2648 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2649     \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2650     \cs_set_nopar:Npn {#3}{#4}
2651     \tl_if_empty:nF {#5}{
2652         \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2653     }
2654     \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2655         \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2656     }
2657 }
2658
2659 \cs_new_protected:Nn \_stex_notation_final: {
2660
2661     \stex_execute_in_module:x {
2662         \_stex_notation_restore_notation:nnnnn
2663         {\l_stex_get_symbol_uri_str}
2664         {\l_stex_notation_suffix_str}
2665         {\l__stex_notation_arity_str}
2666     }
2667     \exp_after:wN \exp_after:wN \exp_after:wN
2668     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2669     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2670   }
2671   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2672 }
2673
2674 \stex_debug:nn{symbols}{
2675   Notation~\l__stex_notation_suffix_str
2676   ~for~\l_stex_get_symbol_uri_str^^J
2677   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2678   Argument~precedences:~
2679   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2680   Notation: \cs_meaning:c {
2681     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2682     \l__stex_notation_suffix_str
2683     _cs
2684   }
2685 }
2686 % HTML annotations
2687 \stex_if_do_html:T {
2688   \stex_annotate_invisible:nnn { notation }
2689   { \l_stex_get_symbol_uri_str } {
2690     \stex_annotate_invisible:nnn { notationfragment }
2691     { \l__stex_notation_suffix_str }{}
2692     \stex_annotate_invisible:nnn { precedence }
2693     { \l__stex_notation_prec_str }{}
2694
2695     \int_zero:N \l_tmpa_int
2696     \str_set_eq:NN \l__stex_notation_remaining_args_str \l_stex_notation_args_str
2697     \tl_clear:N \l_tmpa_tl
2698     \int_step_inline:nn { \l__stex_notation_arity_str }{
2699       \int_incr:N \l_tmpa_int
2700       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2701       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2702       \str_if_eq:VnTF \l_tmpb_str a {
2703         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2704           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2705           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2706         } }
2707       }{
2708         \str_if_eq:VnTF \l_tmpb_str B {
2709           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2710             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2711             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2712           } }
2713         }{
2714           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2715             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2716           } }
2717       }
2718     }
2719   }
2720   \stex_annotate_invisible:nnn { notationcomp }{}{
2721     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2722     $ \exp_args:Nno \use:nn { \use:c {

```



```

2723         stex_notation_ \l_stex_current_symbol_str
2724         \c_hash_str \l__stex_notation_suffix_str _cs
2725     } } { \l_tmpa_tl } $
2726   }
2727 }
2728 }
2729 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2730 \keys_define:nn { stex / setnotation } {
2731   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2732   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2733   unknown .code:n      = \str_set:Nx
2734     \l__stex_notation_variant_str \l_keys_key_str
2735 }
2736
2737 \cs_new_protected:Nn \stex_setnotation_args:n {
2738   % \str_clear:N \l__stex_notation_lang_str
2739   \str_clear:N \l__stex_notation_variant_str
2740   \keys_set:nn { stex / setnotation } { #1 }
2741 }
2742
2743 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2744   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2745     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2746     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2747   }
2748 }
2749
2750 \cs_new_protected:Nn \stex_setnotation:n {
2751   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2752     { \l__stex_notation_variant_str }{
2753     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2754     \stex_debug:nn {notations}{
2755       Setting~default~notation~
2756       {\l__stex_notation_variant_str }~for~
2757       #1 \\
2758       \expandafter\meaning\csname
2759       l_stex_symdecl_#1_notations\endcsname
2760     }
2761   }{
2762     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2763   }
2764 }
2765
2766 \NewDocumentCommand \setnotation {m m} {
2767   \stex_get_symbol:n { #1 }
2768   \stex_setnotation_args:n { #2 }
2769   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2770   \stex_smsmode_do:\ignorespacesandpars
2771 }
2772

```

```

2773 \cs_new_protected:Nn \stex_copy_notations:nn {
2774   \stex_debug:nn {notations}{
2775     Copying~notations~from~#2~to~#1\\
2776     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2777   }
2778   \tl_clear:N \l_tmpa_tl
2779   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2780     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2781   }
2782   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2783     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2784     \edef \l_tmpa_tl {
2785       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2786       \exp_after:wN\exp_after:wN\exp_after:wN {
2787         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2788       }
2789     }
2790
2791     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2792     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2793     \exp_after:wN { \l_tmpa_tl }
2794
2795     \edef \l_tmpa_tl {
2796       \exp_after:wN \exp_not:n \exp_after:wN {
2797         \l_tmpa_tl {##### 1}{##### 2}
2798       }
2799     }
2800
2801     \stex_execute_in_module:x {
2802       \__stex_notation_restore_notation:nnnnn
2803       {#1}{##1}
2804       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2805       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2806       {
2807         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2808           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2809             }
2810         }
2811       }
2812     }
2813   }
2814
2815   \NewDocumentCommand \copynotation {m m} {
2816     \stex_get_symbol:n { #1 }
2817     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2818     \stex_get_symbol:n { #2 }
2819     \exp_args:Noo
2820     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2821     \stex_smsmode_do:\ignorespacesandpars
2822   }
2823

```

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

\symdef

```
2824 \keys_define:nn { stex / symdef } {
2825   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2826   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2827   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2828   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2829   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2830   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2831   op        .tl_set:N    = \l__stex_notation_op_tl ,
2832   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2833   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2834   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2835   assoc     .choices:nn =
2836     {bin,binl,binr,pre,conj,pwconj}
2837     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},
2838   unknown   .code:n      = \str_set:Nx
2839     \l__stex_notation_variant_str \l_keys_key_str
2840 }
2841
2842 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2843   \str_clear:N \l_stex_symdecl_name_str
2844   \str_clear:N \l_stex_symdecl_args_str
2845   \str_clear:N \l_stex_symdecl_assoc_type_str
2846   \str_clear:N \l_stex_symdecl_reorder_str
2847   \bool_set_false:N \l_stex_symdecl_local_bool
2848   \tl_clear:N \l_stex_symdecl_type_tl
2849   \tl_clear:N \l_stex_symdecl_definiens_tl
2850   % \str_clear:N \l__stex_notation_lang_str
2851   \str_clear:N \l__stex_notation_variant_str
2852   \str_clear:N \l__stex_notation_prec_str
2853   \tl_clear:N \l__stex_notation_op_tl
2854
2855   \keys_set:nn { stex / symdef } { #1 }
2856 }
2857
2858 \NewDocumentCommand \symdef { m O{} } {
2859   \__stex_notation_symdef_args:n { #2 }
2860   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2861   \stex_symdecl_do:n { #1 }
2862   \tl_set:Nn \l_stex_notation_after_do_tl {
2863     \__stex_notation_final:
2864     \stex_smsmode_do:\ignorespacesandpars
2865   }
2866   \str_set:Nx \l_stex_get_symbol_uri_str {
2867     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2868   }
2869   \exp_args:Nx \stex_notation_do:nnnnn
2870     { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2871     { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2872     { \l__stex_notation_variant_str }
2873     { \l__stex_notation_prec_str }
2874 }
2875 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3 Variables

```

2876 <@@=stex_variables>
2877
2878 \keys_define:nn { stex / vardef } {
2879   name      .str_set_x:N = \l__stex_variables_name_str ,
2880   args      .str_set_x:N = \l__stex_variables_args_str ,
2881   type      .tl_set:N    = \l__stex_variables_type_tl ,
2882   def       .tl_set:N    = \l__stex_variables_def_tl ,
2883   op        .tl_set:N    = \l__stex_variables_op_tl ,
2884   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2885   assoc     .choices:nn =
2886     {bin,binl,binr,pre,conj,pwconj}
2887     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2888   bind      .choices:nn =
2889     {forall,exists}
2890     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2891 }
2892
2893 \cs_new_protected:Nn \__stex_variables_args:n {
2894   \str_clear:N \l__stex_variables_name_str
2895   \str_clear:N \l__stex_variables_args_str
2896   \str_clear:N \l__stex_variables_prec_str
2897   \str_clear:N \l__stex_variables_assoctype_str
2898   \str_clear:N \l__stex_variables_bind_str
2899   \tl_clear:N \l__stex_variables_type_tl
2900   \tl_clear:N \l__stex_variables_def_tl
2901   \tl_clear:N \l__stex_variables_op_tl
2902
2903   \keys_set:nn { stex / vardef } { #1 }
2904 }
2905
2906 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2907   \__stex_variables_args:n {#2}
2908   \str_if_empty:NT \l__stex_variables_name_str {
2909     \str_set:Nx \l__stex_variables_name_str { #1 }
2910   }
2911   \prop_clear:N \l_tmpa_prop
2912   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2913
2914   \int_zero:N \l_tmpb_int
2915   \bool_set_true:N \l_tmpa_bool
2916   \str_map_inline:Nn \l__stex_variables_args_str {
2917     \token_case_meaning:NnF ##1 {
2918       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2919       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2920       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2921       {\tl_to_str:n a} {
2922         \bool_set_false:N \l_tmpa_bool
2923         \int_incr:N \l_tmpb_int
2924       }

```

```

2925     {\tl_to_str:n B} {
2926       \bool_set_false:N \l_tmpa_bool
2927       \int_incr:N \l_tmpb_int
2928     }
2929   }{
2930     \msg_error:nxxx{stex}{error/wrongargs}{
2931       variable~\l__stex_variables_name_str
2932     }{##1}
2933   }
2934 }
2935 \bool_if:NTF \l_tmpa_bool {
2936   % possibly numeric
2937   \str_if_empty:NTF \l__stex_variables_args_str {
2938     \prop_put:Nnn \l_tmpa_prop { args } {}
2939     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2940   }{
2941     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2942     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2943     \str_clear:N \l_tmpa_str
2944     \int_step_inline:nn \l_tmpa_int {
2945       \str_put_right:Nn \l_tmpa_str i
2946     }
2947     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2948     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2949   }
2950 } {
2951   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2952   \prop_put:Nnx \l_tmpa_prop { arity }
2953     { \str_count:N \l__stex_variables_args_str }
2954 }
2955 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2956 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2957
2958 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2959
2960 \tl_if_empty:NF \l__stex_variables_op_tl {
2961   \cs_set:cpx {
2962     stex_var_op_notation_\l__stex_variables_name_str_cs
2963   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2964 }
2965
2966 \tl_set:Nn \l_stex_notation_after_do_tl {
2967   \exp_args:Nne \use:nn {
2968     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2969     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2970   } {{
2971     \exp_after:wN \exp_after:wN \exp_after:wN
2972     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2973     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2974   }}
2975   \stex_if_do_html:T {
2976     \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2977       \stex_annotate_invisible:nnn { precedence }
2978       { \l__stex_variables_prec_str }{}

```

```

2979 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l
2980 \stex_annotate_invisible:nnn{args}{}}{\l__stex_variables_args_str }
2981 \stex_annotate_invisible:nnn{macroname}{#1}{}
2982 \tl_if_empty:NF \l__stex_variables_def_tl {
2983 \stex_annotate_invisible:nnn{definiens}{}
2984 {\$ \l__stex_variables_def_tl$}
2985 }
2986 \str_if_empty:NF \l__stex_variables_assoctype_str {
2987 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
2988 }
2989 \str_if_empty:NF \l__stex_variables_bind_str {
2990 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
2991 }
2992 \int_zero:N \l_tmpa_int
2993 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2994 \tl_clear:N \l_tmpa_tl
2995 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2996 \int_incr:N \l_tmpa_int
2997 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2998 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2999 \str_if_eq:VnTF \l_tmpb_str a {
3000 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3001 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3002 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3003 } }
3004 }{
3005 \str_if_eq:VnTF \l_tmpb_str B {
3006 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3007 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3008 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3009 } }
3010 }{
3011 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3012 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3013 } }
3014 }
3015 }
3016 }
3017 \stex_annotate_invisible:nnn { notationcomp }{}{
3018 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3019 $ \exp_args:Nno \use:nn { \use:c {
3020 stex_var_notation_\l__stex_variables_name_str_cs
3021 } } { \l_tmpa_tl } $
3022 }
3023 }
3024 }\ignorespacesandpars
3025 }
3026
3027 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3028 }
3029
3030 \cs_new:Nn \_stex_reset:N {
3031 \tl_if_exist:NTF #1 {
3032 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```

```

3033 }{
3034 \let \exp_not:N #1 \exp_not:N \undefined
3035 }
3036 }
3037
3038 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3039 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3040 \exp_args:Nnx \use:nn {
3041 % TODO
3042 \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3043 #2
3044 }
3045 }{
3046 \stex_reset:N \varnot
3047 \stex_reset:N \vartype
3048 \stex_reset:N \vardefi
3049 }
3050 }
3051
3052 \NewDocumentCommand \vardef { s } {
3053 \IfBooleanTF#1 {
3054 \__stex_variables_do_complex:nn
3055 }{
3056 \__stex_variables_do_simple:nnn
3057 }
3058 }
3059
3060 \NewDocumentCommand \svar { 0{} m }{
3061 \tl_if_empty:nTF {#1}{
3062 \str_set:Nn \l_tmpa_str { #2 }
3063 }{
3064 \str_set:Nn \l_tmpa_str { #1 }
3065 }
3066 \stex_term_omv:nn {
3067 var://\l_tmpa_str
3068 }{
3069 \exp_args:Nnx \use:nn {
3070 \def\comp{\_varcomp}
3071 \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3072 \comp{ #2 }
3073 }{
3074 \stex_reset:N \comp
3075 \stex_reset:N \l_stex_current_symbol_str
3076 }
3077 }
3078 }
3079
3080
3081
3082 \keys_define:nn { stex / varseq } {
3083 name .str_set_x:N = \l__stex_variables_name_str ,
3084 args .int_set:N = \l__stex_variables_args_int ,
3085 type .tl_set:N = \l__stex_variables_type_tl ,
3086 mid .tl_set:N = \l__stex_variables_mid_tl ,

```

```

3087   bind      .choices:nn      =
3088           {forall,exists}
3089           {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3090   }
3091
3092   \cs_new_protected:Nn \__stex_variables_seq_args:n {
3093     \str_clear:N \l__stex_variables_name_str
3094     \int_set:Nn \l__stex_variables_args_int 1
3095     \tl_clear:N \l__stex_variables_type_tl
3096     \str_clear:N \l__stex_variables_bind_str
3097
3098     \keys_set:nn { stex / varseq } { #1 }
3099   }
3100
3101   \NewDocumentCommand \varseq {m O{}} m m m){
3102     \__stex_variables_seq_args:n { #2 }
3103     \str_if_empty:NT \l__stex_variables_name_str {
3104       \str_set:Nx \l__stex_variables_name_str { #1 }
3105     }
3106     \prop_clear:N \l_tmpa_prop
3107     \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3108
3109     \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3110     \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3111       \msg_error:nnxx{stex}{error/seqlength}
3112       {\int_use:N \l__stex_variables_args_int}
3113       {\seq_count:N \l_tmpa_seq}
3114     }
3115     \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3116     \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3117       \msg_error:nnxx{stex}{error/seqlength}
3118       {\int_use:N \l__stex_variables_args_int}
3119       {\seq_count:N \l_tmpb_seq}
3120     }
3121     \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3122     \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3123
3124     \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3125     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3126
3127     \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3128     \int_step_inline:nn \l__stex_variables_args_int {
3129       \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3130     }
3131     \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3132     \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3133     \tl_if_empty:NF \l__stex_variables_mid_tl {
3134       \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3135       \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3136     }
3137     \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3138     \int_step_inline:nn \l__stex_variables_args_int {
3139       \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3140     }

```



```

3141 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3142 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3143
3144
3145 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3146
3147 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3148
3149 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3150
3151 \int_step_inline:nn \l__stex_variables_args_int {
3152   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3153     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3154   }}
3155 }
3156
3157 \tl_set:Nx \l_tmpa_tl {
3158   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3159     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3160   }
3161 }
3162
3163 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3164
3165 \exp_args:Nno \use:nn {
3166   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3167   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3168
3169   \stex_debug:nn{sequences}{New~Sequence:~
3170     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3171     \prop_to_keyval:N \l_tmpa_prop
3172   }
3173   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3174     \tl_if_empty:NF \l__stex_variables_type_tl {
3175       \stex_annotate:nnn {type}{\}{\seqtype\l__stex_variables_type_tl$}
3176     }
3177     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3178     \str_if_empty:NF \l__stex_variables_bind_str {
3179       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3180     }
3181   }}
3182
3183   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3184   \ignorespacesandpars
3185 }
3186
3187 </package>

```

Chapter 30

STEX -Terms Implementation

```
3188 <*package>
3189
3190 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3191
3192 <@@=stex_terms>
3193
3194   Warnings and error messages
3195   \msg_new:nnn{stex}{error/nonotation}{
3196     Symbol~#1~invoked,~but~has~no~notation#2!
3197   }
3198   \msg_new:nnn{stex}{error/notationarg}{
3199     Error~in~parsing~notation~#1
3200   }
3201   \msg_new:nnn{stex}{error/noop}{
3202     Symbol~#1~has~no~operator~notation~for~notation~#2
3203   }
3204   \msg_new:nnn{stex}{error/notallowed}{
3205     Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3206   }
3207   \msg_new:nnn{stex}{error/doubleargument}{
3208     Argument~#1~of~symbol~#2~already~assigned
3209   }
3210   \msg_new:nnn{stex}{error/overarity}{
3211     Argument~#1~invalid~for~symbol~#2~with~arity~#3
3212   }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3212
3213
3214 \bool_new:N \l_stex_allow_semantic_bool
3215 \bool_set_true:N \l_stex_allow_semantic_bool
3216
```

```

3217 \cs_new_protected:Nn \stex_invoke_symbol:n {
3218   \bool_if:NTF \l_stex_allow_semantic_bool {
3219     \str_if_eq:eeF {
3220       \prop_item:cn {
3221         l_stex_symdecl_#1_prop
3222       }{ deprecate }
3223     }{}{
3224       \msg_warning:nxx{stex}{warning/deprecated}{
3225         Symbol~#1
3226       }{
3227         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3228       }
3229     }
3230     \if_mode_math:
3231       \exp_after:wN \__stex_terms_invoke_math:n
3232     \else:
3233       \exp_after:wN \__stex_terms_invoke_text:n
3234     \fi: { #1 }
3235   }{
3236     \msg_error:nxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3237   }
3238 }
3239
3240 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3241   \peek_charcode_remove:NTF ! {
3242     \__stex_terms_invoke_op_custom:nn {#1}
3243   }{
3244     \__stex_terms_invoke_custom:nn {#1}
3245   }
3246 }
3247
3248 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3249   \peek_charcode_remove:NTF ! {
3250     % operator
3251     \peek_charcode_remove:NTF * {
3252       % custom op
3253       \__stex_terms_invoke_op_custom:nn {#1}
3254     }{
3255       % op notation
3256       \peek_charcode:NTF [ {
3257         \__stex_terms_invoke_op_notation:nw {#1}
3258       }{
3259         \__stex_terms_invoke_op_notation:nw {#1}[]
3260       }
3261     }
3262   }{
3263     \peek_charcode_remove:NTF * {
3264       \__stex_terms_invoke_custom:nn {#1}
3265       % custom
3266     }{
3267       % normal
3268       \peek_charcode:NTF [ {
3269         \__stex_terms_invoke_notation:nw {#1}
3270       }{

```

```

3271     \__stex_terms_invoke_notation:nw {#1}[]
3272   }
3273 }
3274 }
3275 }
3276
3277
3278 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3279   \exp_args:Nnx \use:nn {
3280     \def\comp{\_comp}
3281     \str_set:Nn \l_stex_current_symbol_str { #1 }
3282     \bool_set_false:N \l_stex_allow_semantic_bool
3283     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3284       \comp{ #2 }
3285     }
3286   }{
3287     \stex_reset:N \comp
3288     \stex_reset:N \l_stex_current_symbol_str
3289     \bool_set_true:N \l_stex_allow_semantic_bool
3290   }
3291 }
3292
3293 \keys_define:nn { stex / terms } {
3294   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3295   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3296   unknown .code:n = \str_set:Nx
3297     \l_stex_notation_variant_str \l_keys_key_str
3298 }
3299
3300 \cs_new_protected:Nn \__stex_terms_args:n {
3301   % \str_clear:N \l_stex_notation_lang_str
3302   \str_clear:N \l_stex_notation_variant_str
3303
3304   \keys_set:nn { stex / terms } { #1 }
3305 }
3306
3307 \cs_new_protected:Nn \stex_find_notation:nn {
3308   \__stex_terms_args:n { #2 }
3309   \seq_if_empty:cTF {
3310     l_stex_symdecl_ #1 _notations
3311   } {
3312     \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3313   } {
3314     \str_if_empty:NTF \l_stex_notation_variant_str {
3315       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3316     }{
3317       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3318         \l_stex_notation_variant_str
3319       }{
3320         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3321       }{
3322         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3323           ~\l_stex_notation_variant_str
3324         }

```

```

3325     }
3326   }
3327 }
3328 }
3329
3330 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3331   \exp_args:Nnx \use:nn {
3332     \def\comp{\_comp}
3333     \str_set:Nn \l_stex_current_symbol_str { #1 }
3334     \stex_find_notation:nn { #1 }{ #2 }
3335     \bool_set_false:N \l_stex_allow_semantic_bool
3336     \cs_if_exist:cTF {
3337       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3338     }{
3339       \_stex_term_oms:nnn { #1 }{
3340         #1 \c_hash_str \l_stex_notation_variant_str
3341       }{
3342         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3343       }
3344     }{
3345       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3346         \cs_if_exist:cTF {
3347           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3348         }{
3349           \tl_set:Nx \stex_symbol_after_invokation_tl {
3350             \_stex_reset:N \comp
3351             \_stex_reset:N \stex_symbol_after_invokation_tl
3352             \_stex_reset:N \l_stex_current_symbol_str
3353             \bool_set_true:N \l_stex_allow_semantic_bool
3354           }
3355           \def\comp{\_comp}
3356           \str_set:Nn \l_stex_current_symbol_str { #1 }
3357           \bool_set_false:N \l_stex_allow_semantic_bool
3358           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3359         }{
3360           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3361             ~\l_stex_notation_variant_str
3362           }
3363         }
3364       }{
3365         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3366       }
3367     }
3368   }{
3369     \_stex_reset:N \comp
3370     \_stex_reset:N \l_stex_current_symbol_str
3371     \bool_set_true:N \l_stex_allow_semantic_bool
3372   }
3373 }
3374
3375 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3376   \stex_find_notation:nn { #1 }{ #2 }
3377   \cs_if_exist:cTF {
3378     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3379 }{
3380   \tl_set:Nx \stex_symbol_after_invokation_tl {
3381     \_stex_reset:N \comp
3382     \_stex_reset:N \stex_symbol_after_invokation_tl
3383     \_stex_reset:N \l_stex_current_symbol_str
3384     \bool_set_true:N \l_stex_allow_semantic_bool
3385   }
3386   \def\comp{\_comp}
3387   \str_set:Nn \l_stex_current_symbol_str { #1 }
3388   \bool_set_false:N \l_stex_allow_semantic_bool
3389   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3390 }{
3391   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3392     ~\l_stex_notation_variant_str
3393   }
3394 }
3395 }
3396
3397 \prop_new:N \l__stex_terms_custom_args_prop
3398
3399 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3400   \exp_args:Nnx \use:nn {
3401     \bool_set_false:N \l_stex_allow_semantic_bool
3402     \def\comp{\_comp}
3403     \str_set:Nn \l_stex_current_symbol_str { #1 }
3404     \prop_clear:N \l__stex_terms_custom_args_prop
3405     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3406     \prop_get:cnN {
3407       l_stex_symdecl_#1 _prop
3408     }{ args } \l_tmpa_str
3409     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3410     \tl_set:Nn \arg { \__stex_terms_arg: }
3411     \str_if_empty:NTF \l_tmpa_str {
3412       \_stex_term oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3413     }{
3414       \str_if_in:NnTF \l_tmpa_str b {
3415         \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3416       }{
3417         \str_if_in:NnTF \l_tmpa_str B {
3418           \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3419         }{
3420           \_stex_term oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3421         }
3422       }
3423     }
3424     % TODO check that all arguments exist
3425   }{
3426     \_stex_reset:N \l_stex_current_symbol_str
3427     \_stex_reset:N \arg
3428     \_stex_reset:N \comp
3429     \_stex_reset:N \l__stex_terms_custom_args_prop
3430     \bool_set_true:N \l_stex_allow_semantic_bool
3431   }
3432 }

```

```

3433
3434 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3435   \tl_if_empty:nTF {#2}{
3436     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3437     \bool_set_true:N \l_tmpa_bool
3438     \bool_do_while:Nn \l_tmpa_bool {
3439       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3440         \int_incr:N \l_tmpa_int
3441       }{
3442         \bool_set_false:N \l_tmpa_bool
3443       }
3444     }
3445   }{
3446     \int_set:Nn \l_tmpa_int { #2 }
3447   }
3448   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3449   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3450     \msg_error:nnxxx{stex}{error/overarity}
3451     {\int_use:N \l_tmpa_int}
3452     {\l_stex_current_symbol_str}
3453     {\str_count:N \l_tmpa_str}
3454   }
3455   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3456   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3457     \bool_lazy_any:nF {
3458       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3459       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3460     }{
3461       \msg_error:nnxx{stex}{error/doubleargument}
3462       {\int_use:N \l_tmpa_int}
3463       {\l_stex_current_symbol_str}
3464     }
3465   }
3466   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3467   \bool_set_true:N \l_stex_allow_semantic_bool
3468   \IfBooleanTF#1{
3469     \stex_annotate_invisible:n { %TODO
3470       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3471     }
3472   }{ %TODO
3473     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3474   }
3475   \bool_set_false:N \l_stex_allow_semantic_bool
3476 }
3477
3478
3479 \cs_new_protected:Nn \_stex_term_arg:nn {
3480   \bool_set_true:N \l_stex_allow_semantic_bool
3481   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3482   \bool_set_false:N \l_stex_allow_semantic_bool
3483 }
3484
3485 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3486   \exp_args:Nnx \use:nn

```

```

3487 { \int_set:Nn \l__stex_terms_downprec { #2 }
3488   \stex_term_arg:nn { #1 }{ #3 }
3489 }
3490 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3491 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

3492 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3493   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3494   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3495   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3496     \expandafter\if\expandafter\relax\noexpand#3
3497     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3498   }else
3499     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3500   \fi
3501   \l_tmpa_tl
3502 }{
3503   \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3504 }
3505 }
3506
3507 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3508   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3509   \str_if_empty:NTF \l_tmpa_str {
3510     \exp_args:Nx \cs_if_eq:NNTF {
3511       \tl_head:N #1
3512     } \stex_invoke_sequence:n {
3513       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3514       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3515       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3516       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3517       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3518         \exp_not:n{\exp_args:Nnx \use:nn} {
3519           \exp_not:n {
3520             \def\comp{\_varcomp}
3521             \str_set:Nn \l_stex_current_symbol_str
3522             } {varseq://\l_tmpa_str}
3523             \exp_not:n{ ##1 }
3524           }{
3525             \exp_not:n {
3526               \stex_reset:N \comp
3527               \stex_reset:N \l_stex_current_symbol_str
3528             }
3529           }
3530         }}}
3531       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3532       \seq_reverse:N \l_tmpa_seq
3533       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3534       \seq_map_inline:Nn \l_tmpa_seq {
3535         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3536           \exp_args:Nno

```



```

3537         \l_tmpa_cs { ##1 } \l_tmpa_tl
3538     }
3539 }
3540 \tl_set:Nx \l_tmpa_tl {
3541     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3542         \exp_args:No \exp_not:n \l_tmpa_tl
3543     }
3544 }
3545 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3546 }{
3547     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3548 }
3549 } {
3550     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3551 }
3552
3553 }
3554
3555 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3556     \clist_set:Nn \l_tmpa_clist{ #2 }
3557     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3558         \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3559     }{
3560         \clist_reverse:N \l_tmpa_clist
3561         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3562         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3563             \exp_args:No \exp_not:n \l_tmpa_tl
3564         }}
3565         \clist_map_inline:Nn \l_tmpa_clist {
3566             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3567                 \exp_args:Nno
3568                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3569             }
3570         }
3571     }
3572     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3573 }

```

(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
3574 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3575 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3576 \int_new:N \l__stex_terms_downprec
3577 \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
```

```
3578 \tl_set:Nn \l__stex_terms_left_bracket_str (
3579 \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

```
3580 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3581   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3582     \bool_set_false:N \l__stex_terms_brackets_done_bool
3583     #2
3584   } {
3585     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3586       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3587         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3588         \dobrackets { #2 }
3589       }
3590     }{ #2 }
3591   }
3592 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
3593 \bool_new:N \l__stex_terms_brackets_done_bool
3594 %\RequirePackage{scalerel}
3595 \cs_new_protected:Npn \dobrackets #1 {
3596   %\ThisStyle{\if D\m@switch
3597   %   \exp_args:Nnx \use:nn
3598   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3599   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3600   % \else
3601   \exp_args:Nnx \use:nn
3602   {
3603     \bool_set_true:N \l__stex_terms_brackets_done_bool
3604     \int_set:Nn \l__stex_terms_downprec \infprec
3605     \l__stex_terms_left_bracket_str
3606     #1
3607   }
3608   {
3609     \bool_set_false:N \l__stex_terms_brackets_done_bool
3610     \l__stex_terms_right_bracket_str
3611     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3612   }
3613   %\fi}
3614 }
```

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

`\withbrackets`

```
3615 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3616   \exp_args:Nnx \use:nn
3617   {
3618     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```

3619 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3620 #3
3621 }
3622 {
3623 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3624 {\l__stex_terms_left_bracket_str}
3625 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3626 {\l__stex_terms_right_bracket_str}
3627 }
3628 }

```

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

`\STEXinvisible`

```

3629 \cs_new_protected:Npn \STEXinvisible #1 {
3630 \stex_annotate_invisible:n { #1 }
3631 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3632 \cs_new_protected:Nn \_stex_term_oms:nnn {
3633 \stex_annotate:nnn{ OMID }{ #2 }{
3634 #3
3635 }
3636 }
3637
3638 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3639 \__stex_terms_maybe_brackets:nn { #3 }{
3640 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3641 }
3642 }

```

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

`_stex_term_math_omv:nn`

```

3643 \cs_new_protected:Nn \_stex_term_omv:nn {
3644 \stex_annotate:nnn{ OMV }{ #1 }{
3645 #2
3646 }
3647 }

```

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

`_stex_term_math_oma:nnnn`

```

3648 \cs_new_protected:Nn \_stex_term_oma:nnn {
3649 \stex_annotate:nnn{ OMA }{ #2 }{
3650 #3
3651 }
3652 }
3653
3654 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3655 \__stex_terms_maybe_brackets:nn { #3 }{
3656 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

3657 }
3658 }

```

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

`_stex_term_math_omb:nnnn`

```

3659 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3660   \stex_annotate:nnn{ OMBIND }{ #2 }{
3661     #3
3662   }
3663 }
3664
3665 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3666   \__stex_terms_maybe_brackets:nn { #3 }{
3667     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3668   }
3669 }

```

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

`\symref`
`\symname`

```

3670 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3671
3672 \keys_define:nn { stex / symname } {
3673   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3674   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3675   root     .tl_set_x:N = \l__stex_terms_root_tl
3676 }
3677
3678 \cs_new_protected:Nn \stex_symname_args:n {
3679   \tl_clear:N \l__stex_terms_post_tl
3680   \tl_clear:N \l__stex_terms_pre_tl
3681   \tl_clear:N \l__stex_terms_root_str
3682   \keys_set:nn { stex / symname } { #1 }
3683 }
3684
3685 \NewDocumentCommand \symref { m m }{
3686   \let\compemph_uri_prev:\compemph@uri
3687   \let\compemph@uri\symrefemph@uri
3688   \STEXsymbol{#1}!{ #2 }
3689   \let\compemph@uri\compemph_uri_prev:
3690 }
3691
3692 \NewDocumentCommand \synonym { 0{} m m }{
3693   \stex_symname_args:n { #1 }
3694   \let\compemph_uri_prev:\compemph@uri
3695   \let\compemph@uri\symrefemph@uri
3696   % TODO
3697   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3698   \let\compemph@uri\compemph_uri_prev:
3699 }
3700
3701 \NewDocumentCommand \symname { 0{} m }{
3702   \stex_symname_args:n { #1 }
3703   \stex_get_symbol:n { #2 }

```

```

3704 \str_set:Nx \l_tmpa_str {
3705   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3706 }
3707 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3708
3709 \let\compemph_uri_prev:\compemph@uri
3710 \let\compemph@uri\symrefemph@uri
3711 \exp_args:NNx \use:nn
3712 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3713   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3714 } }
3715 \let\compemph@uri\compemph_uri_prev:
3716 }
3717
3718 \NewDocumentCommand \Symname { 0{} m }{
3719   \stex_symname_args:n { #1 }
3720   \stex_get_symbol:n { #2 }
3721   \str_set:Nx \l_tmpa_str {
3722     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3723   }
3724   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3725   \let\compemph_uri_prev:\compemph@uri
3726   \let\compemph@uri\symrefemph@uri
3727   \exp_args:NNx \use:nn
3728   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3729     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3730     \l__stex_terms_post_tl
3731   } }
3732   \let\compemph@uri\compemph_uri_prev:
3733 }

```

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

30.3 Notation Components

```

3734 <@@=stex_notationcomps>

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

3735 \cs_new_protected:Npn \_comp #1 {
3736   \str_if_empty:NF \l_stex_current_symbol_str {
3737     \stex_html_backend:TF {
3738       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3739     }{
3740       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3741     }
3742   }
3743 }
3744
3745 \cs_new_protected:Npn \_varcomp #1 {
3746   \str_if_empty:NF \l_stex_current_symbol_str {
3747     \stex_html_backend:TF {
3748       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3749     }{
3750       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3751     }
3752   }
3753 }
3754
3755 \def\comp{\_comp}
3756
3757 \cs_new_protected:Npn \compemph@uri #1 #2 {
3758   \compemph{ #1 }
3759 }
3760
3761
3762 \cs_new_protected:Npn \compemph #1 {
3763   #1
3764 }
3765
3766 \cs_new_protected:Npn \defemph@uri #1 #2 {
3767   \defemph{#1}
3768 }
3769
3770 \cs_new_protected:Npn \defemph #1 {
3771   \textbf{#1}
3772 }
3773
3774 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3775   \symrefemph{#1}
3776 }
3777
3778 \cs_new_protected:Npn \symrefemph #1 {
3779   \emph{#1}
3780 }
3781
3782 \cs_new_protected:Npn \varemp@uri #1 #2 {
3783   \varemp{#1}
3784 }
3785
3786 \cs_new_protected:Npn \varemp #1 {
3787   #1
3788 }

```

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

\ellipses

```

3789 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3790 \bool_new:N \l_stex_inarray_bool
\parrayline 3791 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3792 \NewDocumentCommand \parray { m m } {
\parraycell 3793   \begin{group}
3794   \bool_set_true:N \l_stex_inarray_bool
3795   \begin{array}{#1}
3796     #2
3797   \end{array}

```

```

3798 \endgroup
3799 }
3800
3801 \NewDocumentCommand \prmatrix { m } {
3802   \begingroup
3803   \bool_set_true:N \l_stex_inarray_bool
3804   \begin{matrix}
3805     #1
3806   \end{matrix}
3807   \endgroup
3808 }
3809
3810 \def \maybepline {
3811   \bool_if:NT \l_stex_inarray_bool {\hline}
3812 }
3813
3814 \def \parrayline #1 #2 {
3815   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3816 }
3817
3818 \def \pmrow #1 { \parrayline{}{ #1 } }
3819
3820 \def \parraylineh #1 #2 {
3821   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3822 }
3823
3824 \def \parraycell #1 {
3825   #1 \bool_if:NT \l_stex_inarray_bool {&}
3826 }

```

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

30.4 Variables

```

3827 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3828 \cs_new_protected:Nn \stex_invoke_variable:n {
3829   \if_mode_math:
3830     \exp_after:wN \__stex_variables_invoke_math:n
3831   \else:
3832     \exp_after:wN \__stex_variables_invoke_text:n
3833   \fi: {#1}
3834 }
3835
3836 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3837   %TODO
3838 }
3839
3840
3841 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3842   \peek_charcode_remove:NTF ! {
3843     \peek_charcode_remove:NTF ! {
3844       \peek_charcode:NTF [ {

```

```

3845     \__stex_variables_invoke_op_custom:nw
3846   }{
3847     % TODO throw error
3848   }
3849   ){
3850     \__stex_variables_invoke_op:n { #1 }
3851   }
3852   ){
3853     \peek_charcode_remove:NTF * {
3854       \__stex_variables_invoke_text:n { #1 }
3855     }{
3856       \__stex_variables_invoke_math_ii:n { #1 }
3857     }
3858   }
3859 }
3860
3861 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3862   \cs_if_exist:cTF {
3863     stex_var_op_notation_ #1 _cs
3864   }{
3865     \exp_args:Nnx \use:nn {
3866       \def\comp{\_varcomp}
3867       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3868       \_stex_term_omv:nn { var://#1 }{
3869         \use:c{stex_var_op_notation_ #1 _cs }
3870       }
3871     }{
3872       \_stex_reset:N \comp
3873       \_stex_reset:N \l_stex_current_symbol_str
3874     }
3875   }{
3876     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3877       \__stex_variables_invoke_math_ii:n {#1}
3878     }{
3879       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3880     }
3881   }
3882 }
3883
3884 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3885   \cs_if_exist:cTF {
3886     stex_var_notation_#1_cs
3887   }{
3888     \tl_set:Nx \stex_symbol_after_invokation_tl {
3889       \_stex_reset:N \comp
3890       \_stex_reset:N \stex_symbol_after_invokation_tl
3891       \_stex_reset:N \l_stex_current_symbol_str
3892       \bool_set_true:N \l_stex_allow_semantic_bool
3893     }
3894     \def\comp{\_varcomp}
3895     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3896     \bool_set_false:N \l_stex_allow_semantic_bool
3897     \use:c{stex_var_notation_#1_cs}
3898   }{

```



```

3899   \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3900 }
3901 }

```

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

30.5 Sequences

```

3902 <@@=stex_sequences>
3903
3904 \cs_new_protected:Nn \stex_invoke_sequence:n {
3905   \peek_charcode_remove:NTF ! {
3906     \stex_term_omv:nn {varseq://#1}{
3907       \exp_args:Nnx \use:nn {
3908         \def\comp{\_varcomp}
3909         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3910         \prop_item:cn{stex_varseq_#1_prop}{notation}
3911       }{
3912         \stex_reset:N \comp
3913         \stex_reset:N \l_stex_current_symbol_str
3914       }
3915     }
3916   }{
3917     \bool_set_false:N \l_stex_allow_semantic_bool
3918     \def\comp{\_varcomp}
3919     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3920     \tl_set:Nx \stex_symbol_after_invokation_tl {
3921       \stex_reset:N \comp
3922       \stex_reset:N \stex_symbol_after_invokation_tl
3923       \stex_reset:N \l_stex_current_symbol_str
3924       \bool_set_true:N \l_stex_allow_semantic_bool
3925     }
3926     \use:c { stex_varseq_#1_cs }
3927   }
3928 }
3929 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3930 <*package>
3931
3932 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3933
3934 Warnings and error messages
3935 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3936   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3937 }
3938 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3939   Symbol~#1~not~assigned~in~interpretmodule~#2
3940 }
3941 \msg_new:nnn{stex}{error/unknownstructure}{
3942   No~structure~#1~found!
3943 }
3944 \msg_new:nnn{stex}{error/unknownfield}{
3945   No~field~#1~in~instance~#2~found!\#3
3946 }
3947 \msg_new:nnn{stex}{error/keyval}{
3948   Invalid~key=value~pair~#1
3949 }
3950 \msg_new:nnn{stex}{error/instantiate/missing}{
3951   Assignments~missing~in~instantiate:~#1
3952 }
3953 \msg_new:nnn{stex}{error/incompatible}{
3954   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3955 }
3956
3957
3958
```

31.1 Imports with modification

```

3959 <@@=stex_copymodule>
3960 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3961   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3962     \tl_set:Nn \l_tmpa_tl { #1 }
3963     \__stex_copymodule_get_symbol_from_cs:
3964   }{
3965     % argument is a string
3966     % is it a command name?
3967     \cs_if_exist:cTF { #1 }{
3968       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3969       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3970       \str_if_empty:NNTF \l_tmpa_str {
3971         \exp_args:Nx \cs_if_eq:NNTF {
3972           \tl_head:N \l_tmpa_tl
3973         } \stex_invoke_symbol:n {
3974           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3975         }{
3976           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3977         }
3978       } {
3979         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3980       }
3981     }{
3982       % argument is not a command name
3983       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3984       % \l_stex_all_symbols_seq
3985     }
3986   }
3987 }
3988
3989 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3990   \str_set:Nn \l_tmpa_str { #1 }
3991   \bool_set_false:N \l_tmpa_bool
3992   \bool_if:NF \l_tmpa_bool {
3993     \tl_set:Nn \l_tmpa_tl {
3994       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3995     }
3996     \str_set:Nn \l_tmpa_str { #1 }
3997     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3998     \seq_map_inline:Nn #2 {
3999       \str_set:Nn \l_tmpb_str { ##1 }
4000       \str_if_eq:eeT { \l_tmpa_str } {
4001         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4002       } {
4003         \seq_map_break:n {
4004           \tl_set:Nn \l_tmpa_tl {
4005             \str_set:Nn \l_stex_get_symbol_uri_str {
4006               ##1
4007             }
4008           }
4009         }
4010       }

```

```

4011     }
4012     \l_tmpa_tl
4013   }
4014 }
4015
4016 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4017   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4018     { \tl_tail:N \l_tmpa_tl }
4019   \tl_if_single:NTF \l_tmpa_tl {
4020     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4021       \exp_after:wN \str_set:Nn \exp_after:wN
4022         \l_stex_get_symbol_uri_str \l_tmpa_tl
4023       \__stex_copymodule_get_symbol_check:n { #1 }
4024     }{
4025       % TODO
4026       % tail is not a single group
4027     }
4028   }{
4029     % TODO
4030     % tail is not a single group
4031   }
4032 }
4033
4034 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4035   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4036     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4037       :~\seq_use:Nn #1 {,~}
4038     }
4039   }
4040 }
4041
4042 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4043   % import module
4044   \stex_import_module_uri:nn { #1 } { #2 }
4045   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4046   \stex_import_require_module:nnnn
4047     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4048     { \l_stex_import_path_str } { \l_stex_import_name_str }
4049
4050   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4051   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4052
4053   % fields
4054   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4055   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4056     \seq_map_inline:cn {c_stex_module_##1_constants}{
4057       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4058         ##1 ? ####1
4059       }
4060     }
4061   }
4062
4063   % setup prop
4064   \seq_clear:N \l_tmpa_seq

```

```

4065 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4066   name      = \l_stex_current_copymodule_name_str ,
4067   module    = \l_stex_current_module_str ,
4068   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4069   includes  = \l_tmpa_seq %,
4070 % fields    = \l_tmpa_seq
4071 }
4072 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4073   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4074 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4075 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4076
4077 \stex_if_do_html:T {
4078   \begin{stex_annotate_env} {#4} {
4079     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4080   }
4081   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4082 }
4083 }
4084
4085 \cs_new_protected:Nn \stex_copymodule_end:n {
4086   % apply to every field
4087   \def \l_tmpa_cs ##1 ##2 {#1}
4088
4089   \tl_clear:N \__stex_copymodule_module_tl
4090   \tl_clear:N \__stex_copymodule_exec_tl
4091
4092   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4093   \seq_clear:N \__stex_copymodule_fields_seq
4094
4095   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4096     \seq_map_inline:cn {c_stex_module_##1_constants}{
4097
4098       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4099       \l_tmpa_cs{##1}{####1}
4100
4101       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4102         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4103         \stex_if_do_html:T {
4104           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4105             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4106           }
4107         }
4108       }{
4109         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4110       }
4111
4112       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4113       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4114       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4115
4116       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4117         \stex_if_do_html:T {
4118           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4119         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4120     }
4121 }
4122 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4123 }
4124
4125 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4126 \tl_put_right:Nx \__stex_copymodule_module_tl {
4127     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4128     \prop_set_from_keyval:cn {
4129         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4130     }{
4131         \prop_to_keyval:N \l_tmpa_prop
4132     }
4133 }
4134
4135 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4136     \stex_if_do_html:T {
4137         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4138             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4139         }
4140     }
4141     \tl_put_right:Nx \__stex_copymodule_module_tl {
4142         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4143             \stex_invoke_symbol:n {
4144                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4145             }
4146         }
4147     }
4148 }
4149
4150 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4151
4152 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4153     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4154 }
4155
4156 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4157     \stex_if_do_html:TF{
4158         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4159     }{
4160         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4161     }
4162 }
4163 }
4164 }
4165
4166
4167 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4168 \tl_put_left:Nx \__stex_copymodule_module_tl {
4169     \prop_set_from_keyval:cn {
4170         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4171     }{
4172         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4173     }
4174   }
4175
4176   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4177     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4178   }
4179
4180   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4181   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4182   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4183
4184   \__stex_copymodule_exec_tl
4185   \stex_if_do_html:T {
4186     \end{stex_annotate_env}
4187   }
4188 }
4189
4190 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4191   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4192   \stex_deactivate_macro:Nn \symdecl {module~environments}
4193   \stex_deactivate_macro:Nn \symdef {module~environments}
4194   \stex_deactivate_macro:Nn \notation {module~environments}
4195   \stex_reactivate_macro:N \assign
4196   \stex_reactivate_macro:N \renamedekl
4197   \stex_reactivate_macro:N \donotcopy
4198   \stex_smsmode_do:
4199 }{
4200   \stex_copymodule_end:n {}
4201 }
4202
4203 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4204   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4205   \stex_deactivate_macro:Nn \symdecl {module~environments}
4206   \stex_deactivate_macro:Nn \symdef {module~environments}
4207   \stex_deactivate_macro:Nn \notation {module~environments}
4208   \stex_reactivate_macro:N \assign
4209   \stex_reactivate_macro:N \renamedekl
4210   \stex_reactivate_macro:N \donotcopy
4211   \stex_smsmode_do:
4212 }{
4213   \stex_copymodule_end:n {
4214     \tl_if_exist:cF {
4215       l__stex_copymodule_copymodule_##1?##2_def_tl
4216     }{
4217       \str_if_eq:eeF {
4218         \prop_item:cn{
4219           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4220       }{ true }{
4221         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4222           ##1?##2
4223         }{\l_stex_current_copymodule_name_str}
4224       }
4225     }
4226   }

```

```

4227 }
4228
4229 \iffalse \begin{stex_annotate_env} \fi
4230 \NewDocumentEnvironment {realization} { 0 } { m } {
4231   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4232   \stex_deactivate_macro:Nn \symdecl {module~environments}
4233   \stex_deactivate_macro:Nn \symdef {module~environments}
4234   \stex_deactivate_macro:Nn \notation {module~environments}
4235   \stex_reactivate_macro:N \donotcopy
4236   \stex_reactivate_macro:N \assign
4237   \stex_smsmode_do:
4238 } {
4239   \stex_import_module_uri:nn { #1 } { #2 }
4240   \tl_clear:N \__stex_copymodule_exec_tl
4241   \tl_set:Nx \__stex_copymodule_module_tl {
4242     \stex_import_require_module:nnnn
4243     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4244     { \l_stex_import_path_str } { \l_stex_import_name_str }
4245   }
4246
4247   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4248     \seq_map_inline:cn {c_stex_module_##1_constants}{
4249       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4250       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4251         \stex_if_do_html:T {
4252           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4253             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4254               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4255             }
4256           }
4257         }
4258         \tl_put_right:Nx \__stex_copymodule_module_tl {
4259           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4260         }
4261       }
4262     }
4263
4264     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4265
4266     \__stex_copymodule_exec_tl
4267     \stex_if_do_html:T {\end{stex_annotate_env}}
4268   }
4269
4270 \NewDocumentCommand \donotcopy { m } {
4271   \str_clear:N \l_stex_import_name_str
4272   \str_set:Nn \l_tmpa_str { #1 }
4273   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4274   \seq_map_inline:Nn \l_stex_all_modules_seq {
4275     \str_set:Nn \l_tmpb_str { ##1 }
4276     \str_if_eq:eeT { \l_tmpa_str } {
4277       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4278     } {
4279       \seq_map_break:n {
4280         \stex_if_do_html:T {

```



```

4281         \stex_if_smsmode:F {
4282             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4283                 \stex_annotate:nnn{domain}{##1}{}}
4284         }
4285     }
4286 }
4287 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4288 }
4289 }
4290 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4291     \str_set:Nn \l_tmpb_str { #####1 }
4292     \str_if_eq:eeT { \l_tmpa_str } {
4293         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4294     } {
4295         \seq_map_break:n {\seq_map_break:n {
4296             \stex_if_do_html:T {
4297                 \stex_if_smsmode:F {
4298                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4299                         \stex_annotate:nnn{domain}{
4300                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4301                         }{}}
4302                 }
4303             }
4304         }
4305         \str_set:Nx \l_stex_import_name_str {
4306             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4307         }
4308     }}
4309 }
4310 }
4311 }
4312 \str_if_empty:NTF \l_stex_import_name_str {
4313     % TODO throw error
4314 }{
4315     \stex_collect_imports:n {\l_stex_import_name_str }
4316     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4317         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4318         \seq_map_inline:cn {c_stex_module_###1_constants}{
4319             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4320             \bool_lazy_any:nT {
4321                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4322                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4323                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4324             }{
4325                 % TODO throw error
4326             }
4327         }
4328     }
4329     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4330     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4331     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4332 }
4333 \stex_smsmode_do:
4334 }

```

```

4335
4336 \NewDocumentCommand \assign { m m }{
4337   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4338   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4339   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4340   \stex_smsmode_do:
4341 }
4342
4343 \keys_define:nn { stex / renamedecl } {
4344   name          .str_set_x:N = \l_stex_renamedecl_name_str
4345 }
4346 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4347   \str_clear:N \l_stex_renamedecl_name_str
4348   \keys_set:nn { stex / renamedecl } { #1 }
4349 }
4350
4351 \NewDocumentCommand \renamedecl { O{} m m }{
4352   \__stex_copymodule_renamedecl_args:n { #1 }
4353   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4354   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4355   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4356   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4357     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4358       \l_stex_get_symbol_uri_str
4359     } }
4360   } {
4361     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4362       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4363       \prop_set_eq:cc {l_stex_symdecl_
4364         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4365         _prop
4366       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4367       \seq_set_eq:cc {l_stex_symdecl_
4368         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4369         _notations
4370       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4371       \prop_put:cnx {l_stex_symdecl_
4372         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4373         _prop
4374       }{ name }{ \l_stex_renamedecl_name_str }
4375       \prop_put:cnx {l_stex_symdecl_
4376         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4377         _prop
4378       }{ module }{ \l_stex_current_module_str }
4379       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4380         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4381       }
4382       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4383         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4384       } }
4385     }
4386   \stex_smsmode_do:
4387 }
4388

```

```

4389 \stex_deactivate_macro:Nn \assign {copymodules}
4390 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4391 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4392
4393

```

31.2 The feature environment

structural@feature

```

4394 <@@=stex_features>
4395
4396 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4397   \stex_if_in_module:F {
4398     \msg_set:nnn{stex}{error/nomodule}{
4399       Structural~Feature~has~to~occur~in~a~module:\\
4400       Feature~#2~of~type~#1\\
4401       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4402     }
4403     \msg_error:nn{stex}{error/nomodule}
4404   }
4405
4406   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4407
4408   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4409
4410   \stex_if_do_html:T {
4411     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4412     \stex_annotate_invisible:nnn{header}{\{ #3 }
4413   }
4414   }{
4415     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4416     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4417     \stex_debug:nn{features}{
4418       Feature: \l_stex_last_feature_str
4419     }
4420     \stex_if_do_html:T {
4421       \end{stex_annotate_env}
4422     }
4423   }

```

31.3 Structure

structure

```

4424 <@@=stex_structures>
4425 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4426   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4427     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4428   }
4429   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4430   {#1}{#2}
4431 }
4432

```

```

4433 \keys_define:nn { stex / features / structure } {
4434   name          .str_set_x:N = \l__stex_structures_name_str ,
4435 }
4436
4437 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4438   \str_clear:N \l__stex_structures_name_str
4439   \keys_set:nn { stex / features / structure } { #1 }
4440 }
4441
4442 \NewDocumentEnvironment{mathstructure}{m O{}}{
4443   \__stex_structures_structure_args:n { #2 }
4444   \str_if_empty:NT \l__stex_structures_name_str {
4445     \str_set:Nx \l__stex_structures_name_str { #1 }
4446   }
4447   \stex_suppress_html:n {
4448     \exp_args:Nx \stex_symdecl_do:nn {
4449       name = \l__stex_structures_name_str ,
4450       def  = {\STEXsymbol{module-type}}{
4451         \stex_term_math_oms:nnnn {
4452           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4453             { ns } ?
4454           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4455             { name } / \l__stex_structures_name_str - structure
4456         }{}{0}{}
4457       }}
4458     }{ #1 }
4459   }
4460   \exp_args:Nnnx
4461   \begin{structural_feature_module}{ structure }
4462     { \l__stex_structures_name_str }{}
4463   \stex_smsmode_do:
4464   }{
4465     \end{structural_feature_module}
4466     \stex_reset_up_to_module:n \l_stex_last_feature_str
4467     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4468     \seq_clear:N \l_tmpa_seq
4469     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4470       \seq_map_inline:cn{c_stex_module_##1_constants}{
4471         \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4472       }
4473     }
4474     \exp_args:Nnno
4475     \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4476     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4477     \stex_add_structure_to_current_module:nn
4478       \l__stex_structures_name_str
4479       \l_stex_last_feature_str
4480
4481     \stex_execute_in_module:x {
4482       \tl_set:cn { #1 }{
4483         \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4484       }
4485     }
4486   }

```

```

4487
4488 \cs_new:Nn \stex_invoke_structure:nn {
4489   \stex_invoke_symbol:n { #1?#2 }
4490 }
4491
4492 \cs_new_protected:Nn \stex_get_structure:n {
4493   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4494     \tl_set:Nn \l_tmpa_tl { #1 }
4495     \__stex_structures_get_from_cs:
4496   }{
4497     \cs_if_exist:cTF { #1 }{
4498       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4499       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4500       \str_if_empty:NNTF \l_tmpa_str {
4501         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4502           \__stex_structures_get_from_cs:
4503         }{
4504           \__stex_structures_get_from_string:n { #1 }
4505         }
4506       }{
4507         \__stex_structures_get_from_string:n { #1 }
4508       }
4509     }{
4510       \__stex_structures_get_from_string:n { #1 }
4511     }
4512   }
4513 }
4514
4515 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4516   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4517     { \tl_tail:N \l_tmpa_tl }
4518   \str_set:Nx \l_tmpa_str {
4519     \exp_after:wN \use_i:nn \l_tmpa_tl
4520   }
4521   \str_set:Nx \l_tmpb_str {
4522     \exp_after:wN \use_ii:nn \l_tmpa_tl
4523   }
4524   \str_set:Nx \l_stex_get_structure_str {
4525     \l_tmpa_str ? \l_tmpb_str
4526   }
4527   \str_set:Nx \l_stex_get_structure_module_str {
4528     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4529   }
4530 }
4531
4532 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4533   \tl_set:Nn \l_tmpa_tl {
4534     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4535   }
4536   \str_set:Nn \l_tmpa_str { #1 }
4537   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4538
4539   \seq_map_inline:Nn \l_stex_all_modules_seq {
4540     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4541 \prop_map_inline:cn {c_stex_module_##1_structures} {
4542   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4543     \prop_map_break:n{\seq_map_break:n{
4544       \tl_set:Nn \l_tmpa_tl {
4545         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4546         \str_set:Nn \l_stex_get_structure_module_str {####2}
4547       }
4548     }}
4549   }
4550 }
4551 }
4552 }
4553 \l_tmpa_tl
4554 }

```

\instantiate

```

4555
4556 \keys_define:nn { stex / instantiate } {
4557   name          .str_set_x:N = \l__stex_structures_name_str
4558 }
4559 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4560   \str_clear:N \l__stex_structures_name_str
4561   \keys_set:nn { stex / instantiate } { #1 }
4562 }
4563
4564 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4565   \beginngroup
4566     \stex_get_structure:n {#3}
4567     \__stex_structures_instantiate_args:n { #2 }
4568     \str_if_empty:NT \l__stex_structures_name_str {
4569       \str_set:Nn \l__stex_structures_name_str { #1 }
4570     }
4571     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4572     \seq_clear:N \l__stex_structures_fields_seq
4573     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4574     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4575       \seq_map_inline:cn {c_stex_module_##1_constants}{
4576         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4577       }
4578     }
4579
4580     \tl_if_empty:nF{#5}{
4581       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4582       \prop_clear:N \l_tmpa_prop
4583       \seq_map_inline:Nn \l_tmpa_seq {
4584         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4585         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4586           \msg_error:nnn{stex}{error/keyval}{##1}
4587         }
4588         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4589         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4590         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4591         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4592         \exp_args:Nxx \str_if_eq:nnF

```

```

4593         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4594         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}{
4595         \msg_error:nnxxxx{stex}{error/incompatible}
4596         {l\_stex_structures_dom_str}
4597         {\prop_item:cn{l_stex_symdecl\_l\_stex_structures_dom_str\_prop}{args}}
4598         {\l_stex_get_symbol_uri_str}
4599         {\prop_item:cn{l_stex_symdecl\_l\_stex_get_symbol_uri_str\_prop}{args}}
4600     }
4601     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4602 }
4603 }
4604
4605 \seq_map_inline:Nn \l\_stex_structures_fields_seq {
4606     \str_set:Nx \l_tmpa_str {field:l\_stex_structures_name_str . \prop_item:cn {l_stex_sy
4607     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4608
4609     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4610     \stex_execute_in_module:x {
4611         \prop_set_from_keyval:cn { l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str_p
4612         name = \l_tmpa_str ,
4613         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4614         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4615         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4616     }
4617     \seq_clear:c {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notations}
4618 }
4619
4620 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4621     \stex_find_notation:nn{##1}{}
4622     \stex_execute_in_module:x {
4623         \seq_put_right:cn {l_stex_symdecl\_l\_stex_current_module_str?\l_tmpa_str\_notation
4624     }
4625
4626     \stex_copy_control_sequence_ii:ccN
4627     {stex_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4628     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4629     \l_tmpa_tl
4630     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4631
4632
4633     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4634         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4635         \stex_execute_in_module:x {
4636             \tl_set:cn
4637             {stex_op_notation\_l\_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_nota
4638             { \exp_args:No \exp_not:n \l_tmpa_cs}
4639         }
4640     }
4641
4642 }
4643
4644 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4645 }
4646

```

```

4647 \stex_execute_in_module:x {
4648   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4649   domain = \l_stex_get_structure_module_str ,
4650   \prop_to_keyval:N \l_tmpa_prop
4651 }
4652 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4653 }
4654 \stex_debug:nn{instantiate}{
4655   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4656   \prop_to_keyval:N \l_tmpa_prop
4657 }
4658 \exp_args:Nxx \stex_symdecl_do:nn {
4659   type={\STEXsymbol{module-type}}{
4660     \stex_term_math_oms:nnnn {
4661       \l_stex_get_structure_module_str
4662     }{}{0}{}
4663   }}
4664 }{\l__stex_structures_name_str}
4665 % {
4666   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4667   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4668   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4669 % }
4670 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4671 \endgroup
4672 \stex_smsmode_do:\ignorespacesandpars
4673 }
4674
4675 \cs_new_protected:Nn \stex_symbol_or_var:n {
4676   \cs_if_exist:cTF{#1}{
4677     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4678     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4679     \str_if_empty:NTF \l_tmpa_str {
4680       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4681       \stex_invoke_variable:n {
4682         \bool_set_true:N \l_stex_symbol_or_var_bool
4683         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4684         \str_set:Nx \l_stex_get_symbol_uri_str {
4685           \exp_after:wN \use:n \l_tmpa_tl
4686         }
4687       }{
4688         \bool_set_false:N \l_stex_symbol_or_var_bool
4689         \stex_get_symbol:n{#1}
4690       }
4691     }{
4692       \__stex_structures_symbolorvar_from_string:n{ #1 }
4693     }
4694   }{
4695     \__stex_structures_symbolorvar_from_string:n{ #1 }
4696   }
4697 }
4698
4699 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4700   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```



```

4701     \bool_set_true:N \l_stex_symbol_or_var_bool
4702     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4703   }{
4704     \bool_set_false:N \l_stex_symbol_or_var_bool
4705     \stex_get_symbol:n{#1}
4706   }
4707 }
4708
4709 \keys_define:nn { stex / varinstantiate } {
4710   name      .str_set_x:N = \l__stex_structures_name_str,
4711   bind      .choices:nn =
4712     {forall,exists}
4713     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4714 }
4715
4716 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4717   \str_clear:N \l__stex_structures_name_str
4718   \str_clear:N \l__stex_structures_bind_str
4719   \keys_set:nn { stex / varinstantiate } { #1 }
4720 }
4721
4722 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4723   \beginingroup
4724     \stex_get_structure:n {#3}
4725     \__stex_structures_varinstantiate_args:n { #2 }
4726     \str_if_empty:NT \l__stex_structures_name_str {
4727       \str_set:Nn \l__stex_structures_name_str { #1 }
4728     }
4729     \stex_if_do_html:TF{
4730       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4731     }{\use:n}
4732     {
4733       \stex_if_do_html:T{
4734         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4735       }
4736       \seq_clear:N \l__stex_structures_fields_seq
4737       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4738       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4739         \seq_map_inline:cn {c_stex_module_##1_constants}{
4740           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4741         }
4742       }
4743       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4744       \prop_clear:N \l_tmpa_prop
4745       \tl_if_empty:nF {#5} {
4746         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4747         \seq_map_inline:Nn \l_tmpa_seq {
4748           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4749           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4750             \msg_error:nnn{stex}{error/keyval}{##1}
4751           }
4752           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4753           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4754           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4755 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4756 \stex_if_do_html:T{
4757   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4758 }
4759 \bool_if:NTF \l_stex_symbol_or_var_bool {
4760   \exp_args:Nxx \str_if_eq:nnF
4761     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4762     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4763     \msg_error:nnxxxx{stex}{error/incompatible}
4764     {\l__stex_structures_dom_str
4765     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4766     {\l_stex_get_symbol_uri_str
4767     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4768   }
4769   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4770 }{
4771   \exp_args:Nxx \str_if_eq:nnF
4772     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4773     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4774     \msg_error:nnxxxx{stex}{error/incompatible}
4775     {\l__stex_structures_dom_str
4776     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4777     {\l_stex_get_symbol_uri_str
4778     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4779   }
4780   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4781 }
4782 }
4783 }
4784 \tl_gclear:N \g__stex_structures_aftergroup_tl
4785 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4786   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_
4787   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4788   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4789     \stex_find_notation:nn{##1}{
4790       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4791       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4792       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}
4793       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4794         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4795         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4796         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}
4797       }
4798     }
4799   }
4800   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4801     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4802       name = \l_tmpa_str ,
4803       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4804       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4805       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4806     }
4807     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4808     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4809         \cs_set_eq:cc {stex_var_op_notation\_l_tmpa_str_cs}
4810         {g__stex_structures_tmpa_op\_l_tmpa_str_cs}
4811     }
4812     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4813 }
4814 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4815     \prop_set_from_keyval:cn {l_stex_varinstance\_l__stex_structures_name_str_prop }{
4816         domain = \l_stex_get_structure_module_str ,
4817         \prop_to_keyval:N \l_tmpa_prop
4818     }
4819     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4820     \tl_set:cn {l_stex_varinstance\_l__stex_structures_name_str_op_tl}{
4821         \exp_args:Nnx \exp_not:N \use:nn {
4822             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4823             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4824                 \exp_not:n{
4825                     \_varcomp{#4}
4826                 }
4827             }
4828         }{
4829             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4830         }
4831     }
4832 }
4833 }
4834 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4835 \aftergroup\g__stex_structures_aftergroup_tl
4836 \endgroup
4837 \stex_smsmode_do:\ignorespacesandpars
4838 }
4839
4840 \cs_new_protected:Nn \stex_invoke_instance:n {
4841     \peek_charcode_remove:NTF ! {
4842         \stex_invoke_symbol:n{#1}
4843     }{
4844         \_stex_invoke_instance:nn {#1}
4845     }
4846 }
4847
4848
4849 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4850     \peek_charcode_remove:NTF ! {
4851         \exp_args:Nnx \use:nn {
4852             \def\comp{\_varcomp}
4853             \use:c{l_stex_varinstance_#1_op_tl}
4854         }{
4855             \_stex_reset:N \comp
4856         }
4857     }{
4858         \_stex_invoke_varinstance:nn {#1}
4859     }
4860 }
4861
4862 \cs_new_protected:Nn \_stex_invoke_instance:nn {

```

```

4863 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4864   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4865 }{
4866   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4867   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4868     \prop_to_keyval:N \l_tmpa_prop
4869   }
4870 }
4871 }
4872
4873 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4874   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4875     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4876     \l_tmpa_tl
4877   }{
4878     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4879   }
4880 }

```

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

`\stex_invoke_structure:nnn`

```

4881 % #1: URI of the instance
4882 % #2: URI of the instantiated module
4883 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4884   \tl_if_empty:nTF{ #3 }{
4885     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4886       c_stex_feature_ #2 _prop
4887     }
4888     \tl_clear:N \l_tmpa_tl
4889     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4890     \seq_map_inline:Nn \l_tmpa_seq {
4891       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4892       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4893       \cs_if_exist:cT {
4894         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4895       }{
4896         \tl_if_empty:NF \l_tmpa_tl {
4897           \tl_put_right:Nn \l_tmpa_tl {,}
4898         }
4899         \tl_put_right:Nx \l_tmpa_tl {
4900           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4901         }
4902       }
4903     }
4904     \exp_args:No \mathstruct \l_tmpa_tl
4905   }{
4906     \stex_invoke_symbol:n{#1/#3}
4907   }
4908 }

```

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

```

4909 </package>

```

Chapter 32

STEX -Statements Implementation

```
4910 <*package>
4911
4912 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4913
4914 <@@=stex_statements>
    Warnings and error messages
4915
\titleemph
4916 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4917 \keys_define:nn {stex / definiendum }{
4918   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4919   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4920   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4921   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4922 }
4923 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4924   \str_clear:N \l__stex_statements_definiendum_root_str
4925   \tl_clear:N \l__stex_statements_definiendum_post_tl
4926   \str_clear:N \l__stex_statements_definiendum_gfa_str
4927   \keys_set:nn { stex / definiendum }{ #1 }
4928 }
4929 \NewDocumentCommand \definiendum { O{} m m } {
4930   \__stex_statements_definiendum_args:n { #1 }
4931   \stex_get_symbol:n { #2 }
4932   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4933   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4934     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4935     \tl_set:Nn \l_tmpa_tl { #3 }
4936   } {
4937     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4938     \tl_set:Nn \l_tmpa_tl {
4939       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4940     }
4941   }
4942 } {
4943   \tl_set:Nn \l_tmpa_tl { #3 }
4944 }
4945
4946 % TODO root
4947 \stex_html_backend:TF {
4948   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4949 } {
4950   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4951 }
4952 }
4953 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4954
4955 \NewDocumentCommand \definame { 0{ } m } {
4956   \__stex_statements_definiendum_args:n { #1 }
4957   % TODO: root
4958   \stex_get_symbol:n { #2 }
4959   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4960   \str_set:Nx \l_tmpa_str {
4961     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4962   }
4963   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4964   \stex_html_backend:TF {
4965     \stex_if_do_html:T {
4966       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4967         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4968       }
4969     }
4970   } {
4971     \exp_args:Nnx \defemph@uri {
4972       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4973     } { \l_stex_get_symbol_uri_str }
4974   }
4975 }
4976 \stex_deactivate_macro:Nn \definame {definition~environments}
4977
4978 \NewDocumentCommand \Definame { 0{ } m } {
4979   \__stex_statements_definiendum_args:n { #1 }
4980   \stex_get_symbol:n { #2 }
4981   \str_set:Nx \l_tmpa_str {
4982     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4983   }
4984   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

4985 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4986 \stex_html_backend:TF {
4987   \stex_if_do_html:T {
4988     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4989       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4990     }
4991   }
4992 } {
4993   \exp_args:Nnx \defemph@uri {
4994     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4995   } { \l_stex_get_symbol_uri_str }
4996 }
4997 }
4998 \stex_deactivate_macro:Nn \Definame {definition-environments}
4999
5000 \NewDocumentCommand \premise { m }{
5001   \stex_annotate:nnn{ premise }{}{ #1 }
5002 }
5003 \NewDocumentCommand \conclusion { m }{
5004   \stex_annotate:nnn{ conclusion }{}{ #1 }
5005 }
5006 \NewDocumentCommand \definiens { 0{} m }{
5007   \str_clear:N \l_stex_get_symbol_uri_str
5008   \tl_if_empty:nF {#1} {
5009     \stex_get_symbol:n { #1 }
5010   }
5011   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5012     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5013       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5014     }{
5015       % TODO throw error
5016     }
5017   }
5018   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5019   {\l_stex_current_module_str}{
5020     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5021   }{true}{
5022     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5023     \exp_args:Nx \stex_add_to_current_module:n {
5024       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5025     }
5026   }
5027 }
5028 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5029 }
5030
5031 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5032 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5033 \stex_deactivate_macro:Nn \definiens {definition~environments}
5034

```

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

`sdefinition`

```

5035
5036 \keys_define:nn {stex / sdefinition }{
5037   type      .str_set_x:N = \sdefinitiontype,
5038   id        .str_set_x:N = \sdefinitionid,
5039   name      .str_set_x:N = \sdefinitionname,
5040   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5041   title     .tl_set:N     = \sdefinitiontitle
5042 }
5043 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5044   \str_clear:N \sdefinitiontype
5045   \str_clear:N \sdefinitionid
5046   \str_clear:N \sdefinitionname
5047   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5048   \tl_clear:N \sdefinitiontitle
5049   \keys_set:nn { stex / sdefinition }{ #1 }
5050 }
5051
5052 \NewDocumentEnvironment{sdefinition}{0{}}{
5053   \__stex_statements_sdefinition_args:n{ #1 }
5054   \stex_reactivate_macro:N \definiendum
5055   \stex_reactivate_macro:N \definame
5056   \stex_reactivate_macro:N \Definame
5057   \stex_reactivate_macro:N \premise
5058   \stex_reactivate_macro:N \definiens
5059   \stex_if_smsmode:F{
5060     \seq_clear:N \l_tmpa_seq
5061     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5062       \tl_if_empty:nF{ ##1 }{
5063         \stex_get_symbol:n { ##1 }
5064         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5065           \l_stex_get_symbol_uri_str
5066         }
5067       }
5068     }
5069     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5070     \exp_args:Nnnx
5071     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {},}}
5072     \str_if_empty:NF \sdefinitiontype {
5073       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5074     }
5075     \str_if_empty:NF \sdefinitionname {
5076       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5077     }
5078     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5079     \tl_clear:N \l_tmpa_tl
5080     \clist_map_inline:Nn \l_tmpa_clist {
5081       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5082         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5083       }
5084     }
5085     \tl_if_empty:NTF \l_tmpa_tl {
5086       \__stex_statements_sdefinition_start:
5087     }{
5088       \l_tmpa_tl

```



```

5089     }
5090   }
5091   \stex_ref_new_doc_target:n \sdefinitionid
5092   \stex_smsmode_do:
5093 }{
5094   \stex_suppress_html:n {
5095     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5096   }
5097   \stex_if_smsmode:F {
5098     \clist_set:No \l_tmpa_clist \sdefinitiontype
5099     \tl_clear:N \l_tmpa_tl
5100     \clist_map_inline:Nn \l_tmpa_clist {
5101       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5102         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5103       }
5104     }
5105     \tl_if_empty:NTF \l_tmpa_tl {
5106       \__stex_statements_sdefinition_end:
5107     }{
5108       \l_tmpa_tl
5109     }
5110     \end{stex_annotate_env}
5111   }
5112 }

```

\stexpatchdefinition

```

5113 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5114   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5115     ~(\sdefinitiontitle)
5116   }~}
5117 }
5118 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5119
5120 \newcommand\stexpatchdefinition[3] [] {
5121   \str_set:Nx \l_tmpa_str{ #1 }
5122   \str_if_empty:NTF \l_tmpa_str {
5123     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5124     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5125   }{
5126     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5127     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5128   }
5129 }

```

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

\inlinedef inline:

```

5130 \keys_define:nn {stex / inlinedef }{
5131   type      .str_set_x:N = \sdefinitiontype,
5132   id        .str_set_x:N = \sdefinitionid,
5133   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5134   name      .str_set_x:N = \sdefinitionname
5135 }
5136 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5137 \str_clear:N \sdefinitiontype
5138 \str_clear:N \sdefinitionid
5139 \str_clear:N \sdefinitionname
5140 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5141 \keys_set:nn { stex / inlinedef }{ #1 }
5142 }
5143 \NewDocumentCommand \inlinedef { 0{} m } {
5144 \begingroup
5145 \__stex_statements_inlinedef_args:n{ #1 }
5146 \stex_reactivate_macro:N \definiendum
5147 \stex_reactivate_macro:N \definame
5148 \stex_reactivate_macro:N \Definame
5149 \stex_reactivate_macro:N \premise
5150 \stex_reactivate_macro:N \definiens
5151 \stex_ref_new_doc_target:n \sdefinitionid
5152 \stex_if_smsmode:TF{\stex_suppress_html:n {
5153 \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5154 }}{
5155 \seq_clear:N \l_tmpa_seq
5156 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5157 \tl_if_empty:nF{ ##1 }{
5158 \stex_get_symbol:n { ##1 }
5159 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5160 \l_stex_get_symbol_uri_str
5161 }
5162 }
5163 }
5164 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5165 \exp_args:Nnx
5166 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5167 \str_if_empty:NF \sdefinitiontype {
5168 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5169 }
5170 #2
5171 \str_if_empty:NF \sdefinitionname {
5172 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5173 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5174 }
5175 }
5176 }
5177 \endgroup
5178 \stex_smsmode_do:
5179 }

```

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

32.2 Assertions

sassertion

```

5180
5181 \keys_define:nn {stex / sassertion }{
5182 type .str_set_x:N = \sassertiontype,
5183 id .str_set_x:N = \sassertionid,

```

```

5184 title .tl_set:N = \sassertiontitle ,
5185 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5186 name .str_set_x:N = \sassertionname
5187 }
5188 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5189 \str_clear:N \sassertiontype
5190 \str_clear:N \sassertionid
5191 \str_clear:N \sassertionname
5192 \clist_clear:N \l__stex_statements_sassertion_for_clist
5193 \tl_clear:N \sassertiontitle
5194 \keys_set:nn { stex / sassertion }{ #1 }
5195 }
5196
5197 %\tl_new:N \g__stex_statements_aftergroup_tl
5198
5199 \NewDocumentEnvironment{sassertion}{0{}}{
5200 \__stex_statements_sassertion_args:n{ #1 }
5201 \stex_reactivate_macro:N \premise
5202 \stex_reactivate_macro:N \conclusion
5203 \stex_if_smsmode:F {
5204 \seq_clear:N \l_tmpa_seq
5205 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5206 \tl_if_empty:nF{ ##1 }{
5207 \stex_get_symbol:n { ##1 }
5208 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5209 \l_stex_get_symbol_uri_str
5210 }
5211 }
5212 }
5213 \exp_args:Nnnx
5214 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5215 \str_if_empty:NF \sassertiontype {
5216 \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5217 }
5218 \str_if_empty:NF \sassertionname {
5219 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5220 }
5221 \clist_set:Nn \l_tmpa_clist \sassertiontype
5222 \tl_clear:N \l_tmpa_tl
5223 \clist_map_inline:Nn \l_tmpa_clist {
5224 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5225 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5226 }
5227 }
5228 \tl_if_empty:NTF \l_tmpa_tl {
5229 \__stex_statements_sassertion_start:
5230 }{
5231 \l_tmpa_tl
5232 }
5233 }
5234 \str_if_empty:NTF \sassertionid {
5235 \str_if_empty:NF \sassertionname {
5236 \stex_ref_new_doc_target:n {}
5237 }

```

```

5238 } {
5239   \stex_ref_new_doc_target:n \sassertionid
5240 }
5241 \stex_smsmode_do:
5242 }{
5243   \str_if_empty:NF \sassertionname {
5244     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5245     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5246   }
5247   \stex_if_smsmode:F {
5248     \clist_set:Nn \l_tmpa_clist \sassertiontype
5249     \tl_clear:N \l_tmpa_tl
5250     \clist_map_inline:Nn \l_tmpa_clist {
5251       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5252         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5253       }
5254     }
5255     \tl_if_empty:NTF \l_tmpa_tl {
5256       __stex_statements_sassertion_end:
5257     }{
5258       \l_tmpa_tl
5259     }
5260     \end{stex_annotate_env}
5261   }
5262 }

```

\stexpatchassertion

```

5263
5264 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5265   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5266     (\sassertiontitle)
5267   }~}
5268 }
5269 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5270
5271 \newcommand\stexpatchassertion[3] [] {
5272   \str_set:Nx \l_tmpa_str{ #1 }
5273   \str_if_empty:NTF \l_tmpa_str {
5274     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5275     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5276   }{
5277     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5278     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5279   }
5280 }

```

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

\inlineass inline:

```

5281 \keys_define:nn {stex / inlineass }{
5282   type      .str_set_x:N = \sassertiontype,
5283   id        .str_set_x:N = \sassertionid,
5284   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5285   name      .str_set_x:N = \sassertionname

```

```

5286 }
5287 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5288   \str_clear:N \sassertiontype
5289   \str_clear:N \sassertionid
5290   \str_clear:N \sassertionname
5291   \clist_clear:N \l__stex_statements_sassertion_for_clist
5292   \keys_set:nn { stex / inlineass }{ #1 }
5293 }
5294 \NewDocumentCommand \inlineass { 0{} m } {
5295   \begingroup
5296   \stex_reactivate_macro:N \premise
5297   \stex_reactivate_macro:N \conclusion
5298   \__stex_statements_inlineass_args:n{ #1 }
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
5307   \stex_if_smsmode:TF{
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   }{
5313     \seq_clear:N \l_tmpa_seq
5314     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5315       \tl_if_empty:nF{ ##1 }{
5316         \stex_get_symbol:n { ##1 }
5317         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5318           \l_stex_get_symbol_uri_str
5319         }
5320       }
5321     }
5322     \exp_args:Nnx
5323     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5324       \str_if_empty:NF \sassertiontype {
5325         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5326       }
5327       #2
5328       \str_if_empty:NF \sassertionname {
5329         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5330         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5331         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5332       }
5333     }
5334   }
5335   \endgroup
5336   \stex_smsmode_do:
5337 }

```

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

32.3 Examples

sexample

```

5338
5339 \keys_define:nn {stex / sexample }{
5340   type      .str_set_x:N = \exampletype,
5341   id        .str_set_x:N = \sexampleid,
5342   title     .tl_set:N     = \sexampletitle,
5343   name      .str_set_x:N = \sexamplename ,
5344   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5345 }
5346 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5347   \str_clear:N \sexampletype
5348   \str_clear:N \sexampleid
5349   \str_clear:N \sexamplename
5350   \tl_clear:N \sexampletitle
5351   \clist_clear:N \l__stex_statements_sexample_for_clist
5352   \keys_set:nn { stex / sexample }{ #1 }
5353 }
5354
5355 \NewDocumentEnvironment{sexample}{0{}}{
5356   \__stex_statements_sexample_args:n{ #1 }
5357   \stex_reactivate_macro:N \premise
5358   \stex_reactivate_macro:N \conclusion
5359   \stex_if_smsmode:F {
5360     \seq_clear:N \l_tmpa_seq
5361     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5362       \tl_if_empty:NF{ ##1 }{
5363         \stex_get_symbol:n { ##1 }
5364         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5365           \l_stex_get_symbol_uri_str
5366         }
5367       }
5368     }
5369     \exp_args:Nnnx
5370     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5371     \str_if_empty:NF \sexampletype {
5372       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5373     }
5374     \str_if_empty:NF \sexamplename {
5375       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5376     }
5377     \clist_set:N \l_tmpa_clist \sexampletype
5378     \tl_clear:N \l_tmpa_tl
5379     \clist_map_inline:Nn \l_tmpa_clist {
5380       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5381         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5382       }
5383     }
5384     \tl_if_empty:NTF \l_tmpa_tl {
5385       \__stex_statements_sexample_start:
5386     }{
5387       \l_tmpa_tl
5388     }

```

```

5389 }
5390 \str_if_empty:NF \sexampleid {
5391   \stex_ref_new_doc_target:n \sexampleid
5392 }
5393 \stex_smsmode_do:
5394 ){
5395   \str_if_empty:NF \sexamplename {
5396     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5397   }
5398   \stex_if_smsmode:F {
5399     \clist_set:Nn \l_tmpa_clist \sexamplotype
5400     \tl_clear:N \l_tmpa_tl
5401     \clist_map_inline:Nn \l_tmpa_clist {
5402       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5403         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5404       }
5405     }
5406     \tl_if_empty:NTF \l_tmpa_tl {
5407       \__stex_statements_sexample_end:
5408     }{
5409       \l_tmpa_tl
5410     }
5411     \end{stex_annotate_env}
5412   }
5413 }

```

\stexpatchexample

```

5414
5415 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5416   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplotype {
5417     (\sexamplotype)
5418   }~}
5419 }
5420 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5421
5422 \newcommand\stexpatchexample[3]{} {
5423   \str_set:Nx \l_tmpa_str{ #1 }
5424   \str_if_empty:NTF \l_tmpa_str {
5425     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5426     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5427   }{
5428     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5429     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5430   }
5431 }

```

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

\inlineex inline:

```

5432 \keys_define:nn {stex / inlineex }{
5433   type      .str_set_x:N = \sexamplotype,
5434   id        .str_set_x:N = \sexampleid,
5435   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5436   name      .str_set_x:N = \sexamplename

```

```

5437 }
5438 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5439   \str_clear:N \sexamplotype
5440   \str_clear:N \sexampleid
5441   \str_clear:N \sexamplename
5442   \clist_clear:N \l__stex_statements_sexample_for_clist
5443   \keys_set:nn { stex / inlineex }{ #1 }
5444 }
5445 \NewDocumentCommand \inlineex { 0{ } m } {
5446   \beginngroup
5447   \stex_reactivate_macro:N \premise
5448   \stex_reactivate_macro:N \conclusion
5449   \__stex_statements_inlineex_args:n{ #1 }
5450   \str_if_empty:NF \sexampleid {
5451     \stex_ref_new_doc_target:n \sexampleid
5452   }
5453   \stex_if_smsmode:TF{
5454     \str_if_empty:NF \sexamplename {
5455       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5456     }
5457   }{
5458     \seq_clear:N \l_tmpa_seq
5459     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5460       \tl_if_empty:NF{ ##1 }{
5461         \stex_get_symbol:n { ##1 }
5462         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5463           \l_stex_get_symbol_uri_str
5464         }
5465       }
5466     }
5467     \exp_args:Nnx
5468     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5469       \str_if_empty:NF \sexamplotype {
5470         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5471       }
5472       #2
5473       \str_if_empty:NF \sexamplename {
5474         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5475         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5476       }
5477     }
5478   }
5479   \endgroup
5480   \stex_smsmode_do:
5481 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5482 \keys_define:nn { stex / sparagraph } {
5483   id          .str_set_x:N    = \sparagraphid ,

```



```

5484 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5485 type .str_set_x:N = \sparagraphtype ,
5486 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5487 from .tl_set:N = \sparagraphfrom ,
5488 to .tl_set:N = \sparagraphto ,
5489 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5490 name .str_set:N = \sparagraphname ,
5491 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5492 }
5493
5494 \cs_new_protected:Nn \stex_sparagraph_args:n {
5495 \tl_clear:N \l_stex_sparagraph_title_tl
5496 \tl_clear:N \sparagraphfrom
5497 \tl_clear:N \sparagraphto
5498 \tl_clear:N \l_stex_sparagraph_start_tl
5499 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5500 \str_clear:N \sparagraphid
5501 \str_clear:N \sparagraphtype
5502 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5503 \str_clear:N \sparagraphname
5504 \keys_set:nn { stex / sparagraph }{ #1 }
5505 }
5506 \newif\if@in@omtext\@in@omtextfalse
5507
5508 \NewDocumentEnvironment {sparagraph} { 0{} } {
5509 \stex_sparagraph_args:n { #1 }
5510 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5511 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5512 }{
5513 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5514 }
5515 \@in@omtexttrue
5516 \stex_if_smsmode:F {
5517 \seq_clear:N \l_tmpa_seq
5518 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5519 \tl_if_empty:NF{ ##1 }{
5520 \stex_get_symbol:n { ##1 }
5521 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5522 \l_stex_get_symbol_uri_str
5523 }
5524 }
5525 }
5526 \exp_args:Nnnx
5527 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5528 \str_if_empty:NF \sparagraphtype {
5529 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}}
5530 }
5531 \str_if_empty:NF \sparagraphfrom {
5532 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}}
5533 }
5534 \str_if_empty:NF \sparagraphto {
5535 \stex_annotate_invisible:nnn{to}{\sparagraphto}{}}
5536 }
5537 \str_if_empty:NF \sparagraphname {

```

```

5538     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{\{
5539   }
5540   \clist_set:No \l_tmpa_clist \sparagraphtype
5541   \tl_clear:N \l_tmpa_tl
5542   \clist_map_inline:Nn \sparagraphtype {
5543     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5544       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5545     }
5546   }
5547   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5548   \tl_if_empty:NTF \l_tmpa_tl {
5549     \__stex_statements_sparagraph_start:
5550   }{
5551     \l_tmpa_tl
5552   }
5553 }
5554 \clist_set:No \l_tmpa_clist \sparagraphtype
5555 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5556 {
5557   \stex_reactivate_macro:N \definiendum
5558   \stex_reactivate_macro:N \definame
5559   \stex_reactivate_macro:N \Definame
5560   \stex_reactivate_macro:N \premise
5561   \stex_reactivate_macro:N \definiens
5562 }
5563 \str_if_empty:NTF \sparagraphid {
5564   \str_if_empty:NTF \sparagraphname {
5565     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5566       \stex_ref_new_doc_target:n {}
5567     }
5568   } {
5569     \stex_ref_new_doc_target:n {}
5570   }
5571 } {
5572   \stex_ref_new_doc_target:n \sparagraphid
5573 }
5574 \exp_args:NNx
5575 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5576   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5577     \tl_if_empty:nF{ ##1 }{
5578       \stex_get_symbol:n { ##1 }
5579       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5580     }
5581   }
5582 }
5583 \stex_smsmode_do:
5584 \ignorespacesandpars
5585 }{
5586   \str_if_empty:NF \sparagraphname {
5587     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5588     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5589   }
5590   \stex_if_smsmode:F {
5591     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5592 \tl_clear:N \l_tmpa_tl
5593 \clist_map_inline:Nn \l_tmpa_clist {
5594   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5595     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5596   }
5597 }
5598 \tl_if_empty:NTF \l_tmpa_tl {
5599   \__stex_statements_sparagraph_end:
5600 }{
5601   \l_tmpa_tl
5602 }
5603 \end{stex_annotate_env}
5604 }
5605 }

```

\stexpatchparagraph

```

5606
5607 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5608   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5609     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5610       \titleemph{\l_stex_sparagraph_title_tl}:~
5611     }
5612   }{
5613     \titleemph{\l_stex_sparagraph_start_tl}~
5614   }
5615 }
5616 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5617
5618 \newcommand\stexpatchparagraph[3] [] {
5619   \str_set:Nx \l_tmpa_str{ #1 }
5620   \str_if_empty:NTF \l_tmpa_str {
5621     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5622     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5623   }{
5624     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5625     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5626   }
5627 }
5628
5629 \keys_define:nn { stex / inlinepara } {
5630   id      .str_set_x:N = \sparagraphid ,
5631   type    .str_set_x:N = \sparagraphtype ,
5632   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5633   from    .tl_set:N    = \sparagraphfrom ,
5634   to      .tl_set:N    = \sparagraphto ,
5635   name    .str_set:N    = \sparagraphname
5636 }
5637 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5638   \tl_clear:N \sparagraphfrom
5639   \tl_clear:N \sparagraphto
5640   \str_clear:N \sparagraphid
5641   \str_clear:N \sparagraphtype
5642   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5643   \str_clear:N \sparagraphname

```

```

5644 \keys_set:nn { stex / inlinepara }{ #1 }
5645 }
5646 \NewDocumentCommand \inlinepara { 0{} m } {
5647   \beginingroup
5648   \__stex_statements_inlinepara_args:n{ #1 }
5649   \clist_set:No \l_tmpa_clist \sparagraphtype
5650   \str_if_empty:NTF \sparagraphid {
5651     \str_if_empty:NTF \sparagraphname {
5652       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5653         \stex_ref_new_doc_target:n {}
5654       }
5655     } {
5656       \stex_ref_new_doc_target:n {}
5657     }
5658   } {
5659     \stex_ref_new_doc_target:n \sparagraphid
5660   }
5661   \stex_if_smsmode:TF{
5662     \str_if_empty:NF \sparagraphname {
5663       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5664     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5665   }
5666 }{
5667   \seq_clear:N \l_tmpa_seq
5668   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5669     \tl_if_empty:nF{ ##1 }{
5670       \stex_get_symbol:n { ##1 }
5671       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5672         \l_stex_get_symbol_uri_str
5673       }
5674     }
5675   }
5676   \exp_args:Nnx
5677   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5678     \str_if_empty:NF \sparagraphtype {
5679       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5680     }
5681     \str_if_empty:NF \sparagraphfrom {
5682       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5683     }
5684     \str_if_empty:NF \sparagraphto {
5685       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5686     }
5687     \str_if_empty:NF \sparagraphname {
5688       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5689     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5690     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5691   }
5692   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5693     \clist_map_inline:Nn \l_tmpa_seq {
5694       \stex_ref_new_sym_target:n {##1}
5695     }
5696   }
5697   #2

```

```

5698     }
5699   }
5700   \endgroup
5701   \stex_smsmode_do:
5702 }
5703

```

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

```

5704 \</package>

```

Chapter 33

The Implementation

```
5705 <*package>
5706 <@@=stex_sproof>
5707
5708 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5709
```

33.1 Proofs

We first define some keys for the proof environment.

```
5710 \keys_define:nn { stex / spf } {
5711   id          .str_set_x:N = \spfid,
5712   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5713   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5714   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5715   type        .str_set_x:N = \spftype,
5716   title       .tl_set:N    = \spftitle,
5717   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5718   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5719   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5720 }
5721 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5722   \str_clear:N \spfid
5723   \tl_clear:N \l__stex_sproof_spf_for_tl
5724   \tl_clear:N \l__stex_sproof_spf_from_tl
5725   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5726   \str_clear:N \spftype
5727   \tl_clear:N \spftitle
5728   \tl_clear:N \l__stex_sproof_spf_continues_tl
5729   \tl_clear:N \l__stex_sproof_spf_functions_tl
5730   \tl_clear:N \l__stex_sproof_spf_method_tl
5731   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5732   \keys_set:nn { stex / spf }{ #1 }
5733 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5734 \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.

```

5735 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5736 \cs_new_protected:Npn \sproofnumber {
5737   \int_set:Nn \l_tmpa_int {1}
5738   \bool_while_do:nn {
5739     \int_compare_p:nNn {
5740       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5741     } > 0
5742   }{
5743     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5744     \int_incr:N \l_tmpa_int
5745   }
5746 }
5747 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5748   \int_set:Nn \l_tmpa_int {1}
5749   \bool_while_do:nn {
5750     \int_compare_p:nNn {
5751       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5752     } > 0
5753   }{
5754     \int_incr:N \l_tmpa_int
5755   }
5756   \int_compare:nNnF \l_tmpa_int = 1 {
5757     \int_decr:N \l_tmpa_int
5758   }
5759   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5760     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5761   }
5762 }
5763
5764 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5765   \int_set:Nn \l_tmpa_int {1}
5766   \bool_while_do:nn {
5767     \int_compare_p:nNn {
5768       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5769     } > 0
5770   }{
5771     \int_incr:N \l_tmpa_int
5772   }
5773   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5774 }
5775
5776 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5777   \int_set:Nn \l_tmpa_int {1}
5778   \bool_while_do:nn {

```

```

5779 \int_compare_p:nNn {
5780 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5781 } > 0
5782 }{
5783 \int_incr:N \l_tmpa_int
5784 }
5785 \int_decr:N \l_tmpa_int
5786 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5787 }

```

\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

```

5788 \def\sproof@box{
5789 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5790 }
5791 \def\sproofend{
5792 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5793 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5794 }
5795 }

```

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

spf@*@kw

```

5796 \def\spf@proofsketch@kw{Proof~Sketch}
5797 \def\spf@proof@kw{Proof}
5798 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5799 \AddToHook{begindocument}{
5800 \ltx@ifpackageloaded{babel}{
5801 \makeatletter
5802 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5803 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5804 \input{sproof-ngerman.ldf}
5805 }
5806 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5807 \input{sproof-finnish.ldf}
5808 }
5809 \clist_if_in:NnT \l_tmpa_clist {french}{
5810 \input{sproof-french.ldf}
5811 }
5812 \clist_if_in:NnT \l_tmpa_clist {russian}{
5813 \input{sproof-russian.ldf}
5814 }
5815 \makeatother
5816 }{}
5817 }

```

spfsketch

```

5818 \newcommand\spfsketch[2][]{
5819 \begin{group}
5820 \let \premise \stex_proof_premise:

```



```

5821 \__stex_sproof_spf_args:n{#1}
5822 \stex_if_smsmode:TF {
5823   \str_if_empty:NF \spfid {
5824     \stex_ref_new_doc_target:n \spfid
5825   }
5826 }{
5827   \seq_clear:N \l_tmpa_seq
5828   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5829     \tl_if_empty:nF{ ##1 }{
5830       \stex_get_symbol:n { ##1 }
5831       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5832         \l_stex_get_symbol_uri_str
5833       }
5834     }
5835   }
5836   \exp_args:Nnx
5837   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5838     \str_if_empty:NF \spftype {
5839       \stex_annotate_invisible:nnn{type}{\spftype}{
5840     }
5841     \clist_set:No \l_tmpa_clist \spftype
5842     \tl_set:Nn \l_tmpa_tl {
5843       \titleemph{
5844         \tl_if_empty:NTF \spftitle {
5845           \spf@proofsketch@kw
5846         }{
5847           \spftitle
5848         }
5849       }:\sim
5850     }
5851     \clist_map_inline:Nn \l_tmpa_clist {
5852       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5853         \tl_clear:N \l_tmpa_tl
5854       }
5855     }
5856     \str_if_empty:NF \spfid {
5857       \stex_ref_new_doc_target:n \spfid
5858     }
5859     \l_tmpa_tl #2 \sproofend
5860   }
5861 }
5862 \endgroup
5863 \stex_smsmode_do:
5864 }
5865

```

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

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

```

5866 \newenvironment{spfeq}[2][ ]{
5867   \__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

```

5868 \let \premise \stex_proof_premise:
5869 \stex_if_smsmode:TF {
5870   \str_if_empty:NF \spfid {
5871     \stex_ref_new_doc_target:n \spfid
5872   }
5873 }{
5874   \seq_clear:N \l_tmpa_seq
5875   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5876     \tl_if_empty:nF{ ##1 }{
5877       \stex_get_symbol:n { ##1 }
5878       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5879         \l_stex_get_symbol_uri_str
5880       }
5881     }
5882   }
5883   \exp_args:Nnnx
5884   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5885   \str_if_empty:NF \spftype {
5886     \stex_annotate_invisible:nnn{type}{\spftype}{}
5887   }
5888
5889   \clist_set:No \l_tmpa_clist \spftype
5890   \tl_clear:N \l_tmpa_tl
5891   \clist_map_inline:Nn \l_tmpa_clist {
5892     \tl_if_exist:cT {\_stex_sproof_spfeq_##1_start:}{
5893       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_start:}}
5894     }
5895     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5896       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5897     }
5898   }
5899   \tl_if_empty:NTF \l_tmpa_tl {
5900     \_stex_sproof_spfeq_start:
5901   }{
5902     \l_tmpa_tl
5903   }{~#2}
5904   \str_if_empty:NF \spfid {
5905     \stex_ref_new_doc_target:n \spfid
5906   }
5907   \begin{displaymath}\begin{array}{rcll}
5908   }
5909   \stex_smsmode_do:
5910 }{
5911   \stex_if_smsmode:F {
5912     \end{array}\end{displaymath}
5913     \clist_set:No \l_tmpa_clist \spftype
5914     \tl_clear:N \l_tmpa_tl
5915     \clist_map_inline:Nn \l_tmpa_clist {
5916       \tl_if_exist:cT {\_stex_sproof_spfeq_##1_end:}{
5917         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_end:}}
5918       }
5919     }
5920     \tl_if_empty:NTF \l_tmpa_tl {
5921       \_stex_sproof_spfeq_end:

```

```

5922     }{
5923       \l_tmpa_tl
5924     }
5925     \end{stex_annotate_env}
5926   }
5927 }
5928
5929 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5930   \titleemph{
5931     \tl_if_empty:NTF \spftitle {
5932       \spf@proof@kw
5933     }{
5934       \spftitle
5935     }
5936   }:
5937 }
5938 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5939
5940 \newcommand\stexpatchspfeq[3] [] {
5941   \str_set:Nx \l_tmpa_str{ #1 }
5942   \str_if_empty:NTF \l_tmpa_str {
5943     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5944     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5945   }{
5946     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5947     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5948   }
5949 }
5950

```

(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.

```

5951 \newenvironment{sproof}[2] []{
5952   \let \premise \stex_proof_premise:
5953   \intarray_gzero:N \l__stex_sproof_counter_intarray
5954   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5955   \__stex_sproof_spf_args:n{#1}
5956   \stex_if_smsmode:TF {
5957     \str_if_empty:NF \spfid {
5958       \stex_ref_new_doc_target:n \spfid
5959     }
5960   }{
5961     \seq_clear:N \l_tmpa_seq
5962     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5963       \tl_if_empty:nF{ ##1 }{
5964         \stex_get_symbol:n { ##1 }
5965         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5966           \l_stex_get_symbol_uri_str
5967         }
5968       }
5969     }

```

```

5970 \exp_args:Nnnx
5971 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5972 \str_if_empty:NF \spftype {
5973   \stex_annotate_invisible:nnn{type}{\spftype}{}
5974 }
5975
5976 \clist_set:No \l_tmpa_clist \spftype
5977 \tl_clear:N \l_tmpa_tl
5978 \clist_map_inline:Nn \l_tmpa_clist {
5979   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5980     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5981   }
5982   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5983     \tl_set:Nn \l_tmpa_tl {\use:n{}}
5984   }
5985 }
5986 \tl_if_empty:NTF \l_tmpa_tl {
5987   \__stex_sproof_sproof_start:
5988 }{
5989   \l_tmpa_tl
5990 }{~#2}
5991 \str_if_empty:NF \spfid {
5992   \stex_ref_new_doc_target:n \spfid
5993 }
5994 \begin{description}
5995 }
5996 \stex_smsmode_do:
5997 }{
5998   \stex_if_smsmode:F{
5999     \end{description}
6000     \clist_set:No \l_tmpa_clist \spftype
6001     \tl_clear:N \l_tmpa_tl
6002     \clist_map_inline:Nn \l_tmpa_clist {
6003       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6004         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6005       }
6006     }
6007     \tl_if_empty:NTF \l_tmpa_tl {
6008       \__stex_sproof_sproof_end:
6009     }{
6010       \l_tmpa_tl
6011     }
6012     \end{stex_annotate_env}
6013   }
6014 }
6015
6016 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6017   \par\noindent\titleemph{
6018     \tl_if_empty:NTF \spftype {
6019       \spf@proof@kw
6020     }{
6021       \spftype
6022     }
6023   }:

```

```

6024 }
6025 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6026
6027 \newcommand\stexpatchproof[3] [] {
6028   \str_set:Nx \l_tmpa_str{ #1 }
6029   \str_if_empty:NTF \l_tmpa_str {
6030     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6031     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6032   }{
6033     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6034     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6035   }
6036 }

```

\spfidea

```

6037 \newcommand\spfidea[2] []{
6038   \__stex_sproof_spf_args:n{#1}
6039   \titleemph{
6040     \tl_if_empty:NTF \spftype {Proof-Idea}{
6041       \spftype
6042     }:
6043   }~#2
6044   \sproofend
6045 }

```

(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

```

6046 \newenvironment{spfstep}[1] []{
6047   \__stex_sproof_spf_args:n{#1}
6048   \stex_if_smsmode:TF {
6049     \str_if_empty:NF \spfid {
6050       \stex_ref_new_doc_target:n \spfid
6051     }
6052   }{
6053     \@in@omtexttrue
6054     \seq_clear:N \l_tmpa_seq
6055     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6056       \tl_if_empty:nF{ ##1 }{
6057         \stex_get_symbol:n { ##1 }
6058         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6059           \l_stex_get_symbol_uri_str
6060         }
6061       }
6062     }
6063     \exp_args:Nnnx
6064     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {},}
6065     \str_if_empty:NF \spftype {
6066       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6067   }
6068   \clist_set:Nn \l_tmpa_clist \spftype
6069   \tl_set:Nn \l_tmpa_tl {
6070     \item[\sproofnumber]
6071     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6072   }
6073   \clist_map_inline:Nn \l_tmpa_clist {
6074     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6075       \tl_clear:N \l_tmpa_tl
6076     }
6077   }
6078   \l_tmpa_tl
6079   \tl_if_empty:NF \spftitle {
6080     {(\titleemph{\spftitle})\enspace}
6081   }
6082   \str_if_empty:NF \spfid {
6083     \stex_ref_new_doc_target:n \spfid
6084   }
6085   }
6086   \stex_smsmode_do:
6087   \ignorespacesandpars
6088 }{
6089   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6090     \__stex_sproof_inc_counter:
6091   }
6092   \stex_if_smsmode:F {
6093     \end{stex_annotate_env}
6094   }
6095 }

```

spfcomment

```

6096 \newenvironment{spfcomment}[1][]{
6097   \__stex_sproof_spf_args:n{#1}
6098   \clist_set:Nn \l_tmpa_clist \spftype
6099   \tl_set:Nn \l_tmpa_tl {
6100     \item[\sproofnumber]
6101     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6102   }
6103   \clist_map_inline:Nn \l_tmpa_clist {
6104     \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6105       \tl_clear:N \l_tmpa_tl
6106     }
6107   }
6108   \l_tmpa_tl
6109 }{
6110   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6111     \__stex_sproof_inc_counter:
6112   }
6113 }

```

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.

```

6114 \newenvironment{subproof}[2][]{
6115   \_stex_sproof_spf_args:n{#1}
6116   \stex_if_smsmode:TF{
6117     \str_if_empty:NF \spfid {
6118       \stex_ref_new_doc_target:n \spfid
6119     }
6120   }{
6121     \seq_clear:N \l_tmpa_seq
6122     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6123       \tl_if_empty:nF{ ##1 }{
6124         \stex_get_symbol:n { ##1 }
6125         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6126           \l_stex_get_symbol_uri_str
6127         }
6128       }
6129     }
6130     \exp_args:Nnnx
6131     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6132     \str_if_empty:NF \spftype {
6133       \stex_annotate_invisible:nnn{type}{\spftype}{}
6134     }
6135
6136     \clist_set:Nn \l_tmpa_clist \spftype
6137     \tl_set:Nn \l_tmpa_tl {
6138       \item[\sproofnumber]
6139       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6140     }
6141     \clist_map_inline:Nn \l_tmpa_clist {
6142       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6143         \tl_clear:N \l_tmpa_tl
6144       }
6145     }
6146     \l_tmpa_tl
6147     \tl_if_empty:NF \spftitle {
6148       {(\titleemph{\spftitle})\enspace}
6149     }
6150     {~#2}
6151     \str_if_empty:NF \spfid {
6152       \stex_ref_new_doc_target:n \spfid
6153     }
6154   }
6155   \_stex_sproof_add_counter:
6156   \stex_smsmode_do:
6157 }{
6158   \_stex_sproof_remove_counter:
6159   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6160     \_stex_sproof_inc_counter:
6161   }
6162   \stex_if_smsmode:F{
6163     \end{stex_annotate_env}
6164   }
6165 }

```

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

```

6166 \newenvironment{spfcases}[2] [] {
6167   \tl_if_empty:nTF{#1}{
6168     \begin{subproof}[method=by-cases]{#2}
6169   }{
6170     \begin{subproof}[#1,method=by-cases]{#2}
6171   }
6172 }{
6173   \end{subproof}
6174 }

```

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

```

6175 \newenvironment{spfcase}[2] [] {
6176   \__stex_sproof_spf_args:n{#1}
6177   \stex_if_smsmode:TF {
6178     \str_if_empty:NF \spfid {
6179       \stex_ref_new_doc_target:n \spfid
6180     }
6181   }{
6182     \seq_clear:N \l_tmpa_seq
6183     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6184       \tl_if_empty:nF{ ##1 }{
6185         \stex_get_symbol:n { ##1 }
6186         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6187           \l_stex_get_symbol_uri_str
6188         }
6189       }
6190     }
6191     \exp_args:Nnnx
6192     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6193     \str_if_empty:NF \spftype {
6194       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6195     }
6196     \clist_set:Nn \l_tmpa_clist \spftype
6197     \tl_set:Nn \l_tmpa_tl {
6198       \item[\sproofnumber]
6199       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6200     }
6201     \clist_map_inline:Nn \l_tmpa_clist {
6202       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6203         \tl_clear:N \l_tmpa_tl
6204       }
6205     }
6206     \l_tmpa_tl
6207     \tl_if_empty:nF{#2}{
6208       \titleemph{#2}:~
6209     }
6210   }
6211   \__stex_sproof_add_counter:
6212   \stex_smsmode_do:
6213 }{
6214   \__stex_sproof_remove_counter:
6215   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6216     \__stex_sproof_inc_counter:

```



```

6217 }
6218 \stex_if_smsmode:F{
6219   \clist_set:No \l_tmpa_clist \spftype
6220   \tl_set:Nn \l_tmpa_tl{\sproofend}
6221   \clist_map_inline:Nn \l_tmpa_clist {
6222     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6223       \tl_clear:N \l_tmpa_tl
6224     }
6225   }
6226   \l_tmpa_tl
6227   \end{stex_annotate_env}
6228 }
6229 }

```

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

```

6230 \newcommand\spfcasesketch[3] [] {
6231   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6232 }

```

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.

```

6233 \keys_define:nn { stex / just }{
6234   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6235   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6236   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6237   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6238 }

```

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

\spfjust

```

6239 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6240 \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.

```

6241 \newcommand\justarg[2] [] {#2}
6242 \</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

```
6243 <*package>
6244
6245 %%%%%%%%%% others.dtx %%%%%%%%%%
6246
6247 <@@=stex_others>
        Warnings and error messages
6248 % None

\MSC Math subject classifier

6249 \NewDocumentCommand \MSC {m} {
6250 % TODO
6251 }

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

6252 \@ifpackageloaded{tikzinput}{
6253 \RequirePackage{stex-tikzinput}
6254 }{}
6255
6256 \bool_if:NT \c_stex_persist_mode_bool {
6257 \input{\jobname.sms}
6258 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6259 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6260 \l_tmpa_str
6261 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6262 \c_stex_mathhub_main_manifest_prop
6263 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6264 }
6265 }

6266 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6267 <*package>
6268 <@@=stex_modules>
6269
6270 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6271
6272 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6273 \begingroup
6274 \stex_module_setup:nn{
6275   ns=\c_stex_metatheory_ns_str,
6276   meta=NONE
6277 }{Metatheory}
6278 \stex_reactivate_macro:N \symdecl
6279 \stex_reactivate_macro:N \notation
6280 \stex_reactivate_macro:N \symdef
6281 \ExplSyntaxOff
6282 \csname stex_suppress_html:n\endcsname{
6283   % is-a (a:A, a \in A, a is an A, etc.)
6284   \symdecl{isa}[args=ai]
6285   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6286   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6287   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6288
6289   % bind (\forall, \Pi, \lambda etc.)
6290   \symdecl{bind}[args=Bi]
6291   \notation{bind}{forall}{\comp\forall #1.;#2}{##1 \comp, ##2}
6292   \notation{bind}{Pi}{\comp\prod_{#1}#2}{##1 \comp, ##2}
6293   \notation{bind}{depfun}{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
6294
6295   % implicit bind
6296   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6297
6298   % dummy variable
6299   \symdecl{dummyvar}
6300   \notation{dummyvar}[underscore]{\comp\_}
6301   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6302 \notation{dummyvar}[dash]{\comp{\rm --}}
6303
6304 %fromto (function space, Hom-set, implication etc.)
6305 \symdecl{fromto}[args=ai]
6306 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6307 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6308
6309 % mapto (lambda etc.)
6310 \symdecl{mapto}[args=Bi]
6311 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6312 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6313 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6314
6315 % function/operator application
6316 \symdecl{apply}[args=ia]
6317 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6318 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6319
6320 % collection of propositions/booleans/truth values
6321 \symdecl{prop}[name=proposition]
6322 \notation{prop}[prop]{\comp{\rm prop}}
6323 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6324
6325 \symdecl{judgmentholds}[args=1]
6326 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6327
6328 % sequences
6329 \symdecl{seqtype}[args=1]
6330 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6331
6332 \symdecl{seqexpr}[args=a]
6333 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6334
6335 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6336 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6337 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6338 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6339 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6340 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6341 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6342 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6343 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6344
6345 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6346 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6347
6348 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6349 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6350 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6351
6352 % letin ("let", local definitions, variable substitution)
6353 \symdecl{letin}[args=bii]
6354 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
6355 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6356 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6357
6358 % structures
6359 \symdecl*{module-type}[args=1]
6360 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6361 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6362 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6363
6364 % objects
6365 \symdecl{object}
6366 \notation{object}{\comp{\mathtt{OBJECT}}}
6367
6368 }
6369
6370 % The following are abbreviations in the sTeX corpus that are left over from earlier
6371 % developments. They will eventually be phased out.
6372
6373 \ExplSyntaxOn
6374 \stex_add_to_current_module:n{
6375   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6376   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6377   \def\livar{\csname sequence-index\endcsname[li]}
6378   \def\uivar{\csname sequence-index\endcsname[ui]}
6379   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6380   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6381 }
6382 \__stex_modules_end_module:
6383 \endgroup
6384 \</package>

```

Chapter 36

Tikzinput Implementation

```
6385 <@@=tikzinput>
6386 <*package>
6387
6388 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6389
6390 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6391 \RequirePackage{l3keys2e}
6392
6393 \keys_define:nn { tikzinput } {
6394   image .bool_set:N = \c_tikzinput_image_bool,
6395   image .default:n = false ,
6396   unknown .code:n = {}
6397 }
6398
6399 \ProcessKeysOptions { tikzinput }
6400
6401 \bool_if:NTF \c_tikzinput_image_bool {
6402   \RequirePackage{graphicx}
6403
6404   \providecommand\usetikzlibrary[]{}
6405   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6406 }{
6407   \RequirePackage{tikz}
6408   \RequirePackage{standalone}
6409
6410   \newcommand \tikzinput [2] [] {
6411     \setkeys{Gin}{#1}
6412     \ifx \Gin@ewidth \Gin@exclamation
6413       \ifx \Gin@eheight \Gin@exclamation
6414         \input { #2 }
6415       \else
6416         \resizebox{!}{ \Gin@eheight }{
6417           \input { #2 }
6418         }
6419       \fi
6420     \else
6421       \ifx \Gin@eheight \Gin@exclamation
6422         \resizebox{ \Gin@ewidth }{!}{
```

```

6423         \input { #2 }
6424     }
6425     \else
6426         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6427             \input { #2 }
6428         }
6429     \fi
6430 \fi
6431 }
6432 }
6433
6434 \newcommand \ctikzinput [2] [] {
6435     \begin{center}
6436         \tikzinput [#1] {#2}
6437     \end{center}
6438 }
6439
6440 \@ifpackageloaded{stex}{
6441     \RequirePackage{stex-tikzinput}
6442 }{}
6443
6444 \</package>
6445 \<stex>
6446 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6447 \RequirePackage{stex}
6448 \RequirePackage{tikzinput}
6449
6450 \newcommand\mhtikzinput[2] []{%
6451     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6452     \stex_in_repository:nn\Gin@mhrepos{
6453         \tikzinput[#1]{\mhp\path{##1}{#2}}
6454     }
6455 }
6456 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6457
6458 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6459     \pgfkeys@spdef\pgf@temp{#1}
6460     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6461     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6462     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6463     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6464     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6465     \catcode'\@=11
6466     \catcode'\|=12
6467     \catcode'\$=3
6468     \pgfutil@InputIfFileExists{#2}{-}{-}
6469     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6470     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6471     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6472 }
6473
6474
6475 \newcommand\libusetikzlibrary[1]{

```

```

6476 \prop_if_exist:NF \l_stex_current_repository_prop {
6477   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6478 }
6479 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6480   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6481 }
6482 \seq_clear:N \l__tikzinput_libinput_files_seq
6483 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6484 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6485
6486 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6487   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6488   \IfFileExists{ \l_tmpa_str }{
6489     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6490   }{
6491     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6492     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6493   }
6494
6495   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6496   \IfFileExists{ \l_tmpa_str }{
6497     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6498   }{
6499
6500   \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6501     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6502   }{
6503     \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6504       \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6505         \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6506       }
6507     }{
6508       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6509     }
6510   }
6511 }
6512 </stex>

```

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

Chapter 37

document-structure.sty Implementation

```
6513 <*package>
6514 <@@=document_structure>
6515 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6516 \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).

```
6517
6518 \keys_define:nn{ document-structure }{
6519   class      .str_set_x:N = \c_document_structure_class_str,
6520   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6521   unknown    .code:n      = {
6522     \PassOptionsToClass{\CurrentOption}{stex}
6523     \PassOptionsToClass{\CurrentOption}{tikzinput}
6524   }
6525   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6526 }
6527 \ProcessKeysOptions{ document-structure }
6528 \str_if_empty:NT \c_document_structure_class_str {
6529   \str_set:Nn \c_document_structure_class_str {article}
6530 }
6531 \str_if_empty:NT \c_document_structure_topsect_str {
6532   \str_set:Nn \c_document_structure_topsect_str {section}
6533 }
```

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

```
6534 \RequirePackage{xspace}
6535 \RequirePackage{comment}
6536 \RequirePackage{stex}
6537 \AddToHook{begindocument}{}
```

```

6538 \ltx@ifpackageloaded{babel}{
6539     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6540     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6541         \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6542     }
6543 }{}
6544 }

```

`\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.

```

6545 \int_new:N \l_document_structure_section_level_int
6546 \str_case:VnF \c_document_structure_topsect_str {
6547     {part}}{
6548         \int_set:Nn \l_document_structure_section_level_int {0}
6549     }
6550     {chapter}{
6551         \int_set:Nn \l_document_structure_section_level_int {1}
6552     }
6553 }{
6554     \str_case:VnF \c_document_structure_class_str {
6555         {book}{
6556             \int_set:Nn \l_document_structure_section_level_int {0}
6557         }
6558         {report}{
6559             \int_set:Nn \l_document_structure_section_level_int {0}
6560         }
6561     }{
6562         \int_set:Nn \l_document_structure_section_level_int {2}
6563     }
6564 }

```

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

```

6565 \def\current@section@level{document}%
6566 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6567 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6568 \cs_new_protected:Npn \skipfragment {

```

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

```

6569 \ifcase\l_document_structure_section_level_int
6570 \or\stepcounter{part}
6571 \or\stepcounter{chapter}
6572 \or\stepcounter{section}
6573 \or\stepcounter{subsection}
6574 \or\stepcounter{subsubsection}
6575 \or\stepcounter{paragraph}
6576 \or\stepcounter{subparagraph}
6577 \fi
6578 }

```

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

blindfragment

```

6579 \newcommand\at@begin@blindsfragment[1]{
6580 \newenvironment{blindfragment}
6581 {
6582 \int_incr:N\l_document_structure_section_level_int
6583 \at@begin@blindsfragment\l_document_structure_section_level_int
6584 }{}

```

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

```

6585 \newcommand\sfragment@nonum[2]{
6586 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6587 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6588 }

```

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

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

```

6589 \newcommand\sfragment@num[2]{
6590 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6591 \@nameuse{#1}{#2}
6592 }{
6593 \cs_if_exist:NTF\rdmeta@sectioning{
6594 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6595 }{
6596 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6597 }
6598 }
6599 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6600 }

```

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

sfragment

```

6601 \keys_define:nn { document-structure / sfragment }{
6602 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6603 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6604 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6605 contributors .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6606 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6607 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6608 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6609 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6610 intro       .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6611 imports     .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6612 loadmodules .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6613 }
6614 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6615   \str_clear:N \l__document_structure_sfragment_id_str
6616   \str_clear:N \l__document_structure_sfragment_date_str
6617   \clist_clear:N \l__document_structure_sfragment_creators_clist
6618   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6619   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6620   \tl_clear:N \l__document_structure_sfragment_type_tl
6621   \tl_clear:N \l__document_structure_sfragment_short_tl
6622   \tl_clear:N \l__document_structure_sfragment_display_tl
6623   \tl_clear:N \l__document_structure_sfragment_imports_tl
6624   \tl_clear:N \l__document_structure_sfragment_intro_tl
6625   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6626   \keys_set:nn { document-structure / sfragment } { #1 }
6627 }

```

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.

```

6628 \newif\if@mainmatter\@mainmattertrue
6629 \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.

```

6630 \keys_define:nn { document-structure / sectioning }{
6631   name .str_set_x:N = \l__document_structure_sect_name_str ,
6632   ref  .str_set_x:N = \l__document_structure_sect_ref_str  ,
6633   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6634   clear .default:n  = {true} ,
6635   num  .bool_set:N = \l__document_structure_sect_num_bool ,
6636   num  .default:n  = {true}
6637 }
6638 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6639   \str_clear:N \l__document_structure_sect_name_str
6640   \str_clear:N \l__document_structure_sect_ref_str
6641   \bool_set_false:N \l__document_structure_sect_clear_bool
6642   \bool_set_false:N \l__document_structure_sect_num_bool
6643   \keys_set:nn { document-structure / sectioning } { #1 }
6644 }
6645 \newcommand\omdoc@sectioning[3][]{
6646   \l__document_structure_sect_args:n {#1 }
6647   \let\omdoc@sect@name\l__document_structure_sect_name_str
6648   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6649   \if@mainmatter% numbering not overridden by frontmatter, etc.
6650     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6651     \sfragment@num{#2}{#3}
6652   }{
6653     \sfragment@nonum{#2}{#3}
6654   }
6655   \def\current@section@level{\omdoc@sect@name}
6656 \else
6657   \sfragment@nonum{#2}{#3}
6658 \fi
6659 }% 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.

```

6660 \newcommand\sfragment@redefine@addtocontents[1]{%
6661   %\edef\__document_structureimport{#1}%
6662   %\@for\@I:=\__document_structureimport\do{%
6663     %\edef\@path{\csname module@\@I @path\endcsname}%
6664     %\@ifundefined{tf@toc}\relax%
6665     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6666     %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6667     %\def\addcontentsline##1##2##3{%
6668       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6669     %\else% hyperref.sty not loaded
6670     %\def\addcontentsline##1##2##3{%
6671       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6672     %\fi
6673   }% 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.

```

6674 \newenvironment{sfragment}[2] []% keys, title
6675 {
6676   \__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.

```

6677   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6678
6679   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6680     \sfragment@redefine@addtocontents{
6681       %\@ifundefined{module@id}\used@modules%
6682       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6683     }
6684   }

```

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

```

6685
6686   \stex_document_title:n { #2 }
6687
6688   \int_incr:N\l__document_structure_section_level_int
6689   \ifcase\l__document_structure_section_level_int
6690     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6691     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6692 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6693 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6694 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6695 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6696 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6697 \fi
6698 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6699 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6700   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6701 }
6702 }% for customization
6703 {}

```

and finally, we localize the sections

```

6704 \newcommand\omdoc@part@kw{Part}
6705 \newcommand\omdoc@chapter@kw{Chapter}
6706 \newcommand\omdoc@section@kw{Section}
6707 \newcommand\omdoc@subsection@kw{Subsection}
6708 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6709 \newcommand\omdoc@paragraph@kw{paragraph}
6710 \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`

```

6711 \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).

```

6712 \cs_if_exist:NTF\frontmatter{
6713   \let\__document_structure_orig_frontmatter\frontmatter
6714   \let\frontmatter\relax
6715 }{
6716   \tl_set:Nn\__document_structure_orig_frontmatter{
6717     \clearpage
6718     \@mainmatterfalse
6719     \pagenumbering{roman}
6720   }
6721 }
6722 \cs_if_exist:NTF\backmatter{
6723   \let\__document_structure_orig_backmatter\backmatter
6724   \let\backmatter\relax
6725 }{
6726   \tl_set:Nn\__document_structure_orig_backmatter{
6727     \clearpage
6728     \@mainmatterfalse

```

```

6729     \pagenumbering{roman}
6730   }
6731 }

```

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.

```

6732 \newenvironment{frontmatter}{
6733   \_document_structure_orig_frontmatter
6734 }{
6735   \cs_if_exist:NTF\mainmatter{
6736     \mainmatter
6737   }{
6738     \clearpage
6739     \@mainmattertrue
6740     \pagenumbering{arabic}
6741   }
6742 }

```

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

```

6743 \newenvironment{backmatter}{
6744   \_document_structure_orig_backmatter
6745 }{
6746   \cs_if_exist:NTF\mainmatter{
6747     \mainmatter
6748   }{
6749     \clearpage
6750     \@mainmattertrue
6751     \pagenumbering{arabic}
6752   }
6753 }

```

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

```

6754 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6755 \def \c__document_structure_document_str{document}
6756 \newcommand\afterprematurestop{}
6757 \def\prematurestop@endsfragment{
6758   \unless\ifx\@currenvir\c__document_structure_document_str
6759     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6760       \expandafter\prematurestop@endsfragment
6761     }
6762   }
6763 \providecommand\prematurestop{
6764   \message{Stopping~sTeX~processing~prematurely}
6765   \prematurestop@endsfragment
6766   \afterprematurestop
6767   \end{document}
6768 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
6769 \RequirePackage{etoolbox}
6770 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6771 \newrobustcmd\useSGvar[1]{%
6772   \@ifundefined{sTeX@Gvar@#1}
6773   {\PackageError{document-structure}
6774     {The sTeX Global variable #1 is undefined}
6775     {set it with \protect\setSGvar}}
6776   \@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.

```
6777 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6778   \@ifundefined{sTeX@Gvar@#1}
6779   {\PackageError{document-structure}
6780     {The sTeX Global variable #1 is undefined}
6781     {set it with \protect\setSGvar}}
6782   {\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.

```
6783 \*cls)
6784 \@@=notesslides}
6785 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6786 \RequirePackage{13keys2e}
6787
6788 \keys_define:nn{notesslides / cls}{
6789   class   .str_set_x:N = \c__notesslides_class_str,
6790   notes   .bool_set:N = \c__notesslides_notes_bool ,
6791   slides  .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6792   docopt  .str_set_x:N = \c__notesslides_docopt_str,
6793   unknown .code:n      = {
6794     \PassOptionsToPackage{\CurrentOption}{document-structure}
6795     \PassOptionsToClass{\CurrentOption}{beamer}
6796     \PassOptionsToPackage{\CurrentOption}{notesslides}
6797     \PassOptionsToPackage{\CurrentOption}{stex}
6798   }
6799 }
6800 \ProcessKeysOptions{ notesslides / cls }
6801
6802 \str_if_empty:NF \c__notesslides_class_str {
6803   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6804 }
6805
6806 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6807   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6808 }
6809 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6810   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6811 }
6812
6813 \RequirePackage{stex}
```

```

6814 \stex_html_backend:T {
6815   \bool_set_true:N\c__notesslides_notes_bool
6816 }
6817
6818 \bool_if:NTF \c__notesslides_notes_bool {
6819   \PassOptionsToPackage{notes=true}{notesslides}
6820 }{
6821   \PassOptionsToPackage{notes=false}{notesslides}
6822 }
6823 \</cls>

```

now we do the same for the notesslides package.

```

6824 \*package>
6825 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6826 \RequirePackage{13keys2e}
6827
6828 \keys_define:nn{notesslides / pkg}{
6829   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6830   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6831   notes            .bool_set:N = \c__notesslides_notes_bool ,
6832   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6833   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6834   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6835   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6836   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
6837   unknown          .code:n      = {
6838     \PassOptionsToClass{\CurrentOption}{stex}
6839     \PassOptionsToClass{\CurrentOption}{tikzinput}
6840   }
6841 }
6842 \ProcessKeysOptions{ notesslides / pkg }
6843
6844 \RequirePackage{stex}
6845 \stex_html_backend:T {
6846   \bool_set_true:N\c__notesslides_notes_bool
6847 }
6848
6849 \newif\ifnotes
6850 \bool_if:NTF \c__notesslides_notes_bool {
6851   \notesttrue
6852 }{
6853   \notesfalse
6854 }
6855

```

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

```

6856 \str_if_empty:NTF \c__notesslides_topsect_str {
6857   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6858 }{
6859   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6860 }
6861 \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6862 \</package>

```

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

```

6863 <*cls>
6864 \bool_if:NTF \c__notesslides_notes_bool {
6865   \str_if_empty:NT \c__notesslides_class_str {
6866     \str_set:Nn \c__notesslides_class_str {article}
6867   }
6868   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6869     {\c__notesslides_class_str}
6870 }{
6871   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6872   \newcounter{Item}
6873   \newcounter{paragraph}
6874   \newcounter{subparagraph}
6875   \newcounter{Hfootnote}
6876 }
6877 \RequirePackage{document-structure}

```

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

```

6878 \RequirePackage{notesslides}
6879 </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.

```

6880 <*package>
6881 \bool_if:NT \c__notesslides_notes_bool {
6882   \RequirePackage{a4wide}
6883   \RequirePackage{marginnote}
6884   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6885   \RequirePackage{mdframed}
6886   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6887   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6888 }
6889 \RequirePackage{stex-tikzinput}
6890 \RequirePackage{etoolbox}
6891 \RequirePackage{amssymb}
6892 \RequirePackage{amsmath}
6893 \RequirePackage{comment}
6894 \RequirePackage{textcomp}
6895 \RequirePackage{url}
6896 \RequirePackage{graphicx}
6897 \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

```

6898 \bool_if:NT \c__notesslides_notes_bool {
6899   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6900 }
6901
6902
6903 \NewDocumentCommand \libusetheme {0{} m} {
6904   \bool_if:NTF \c__notesslides_notes_bool {
6905     \libusepackage[#1]{beamernotestheme#2}
6906   }{
6907     \libusepackage[#1]{beamertheme#2}
6908   }
6909 }

```

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

```

6910 \newcounter{slide}
6911 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6912 \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.

```

6913 \bool_if:NTF \c__notesslides_notes_bool {
6914   \renewenvironment{note}{\ignorespaces}{}
6915 }{
6916   \excludecomment{note}
6917 }

```

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.

```

6918 \bool_if:NT \c__notesslides_notes_bool {
6919   \newlength{\slideframewidth}
6920   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6921 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6922   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6923     \bool_set_true:N #1
6924   }{
6925     \bool_set_false:N #1
6926   }
6927 }
6928 \keys_define:nn{notesslides / frame}{
6929   label .str_set_x:N = \l__notesslides_frame_label_str,
6930   allowframebreaks .code:n = {
6931     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6932   },
6933   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.

```

6934     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6935 },
6936 fragile .code:n = {
6937     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6938 },
6939 shrink .code:n = {
6940     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6941 },
6942 squeeze .code:n = {
6943     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6944 },
6945 t .code:n = {
6946     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6947 },
6948 }
6949 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6950     \str\_clear:N \l\_notesslides\_frame\_label\_str
6951     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6952     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6953     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6954     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6955     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6956     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6957     \keys\_set:nn { notesslides / frame }{ #1 }
6958 }

```

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

```

6959 \renewenvironment{frame}[1][]{
6960     \_notesslides\_frame\_args:n{#1}
6961     \sffamily
6962     \stepcounter{slide}
6963     \def\@currentlabel{\theslide}
6964     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6965         \label{\l\_notesslides\_frame\_label\_str}
6966     }

```

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

```

6967     \def\itemize@level{outer}
6968     \def\itemize@outer{outer}
6969     \def\itemize@inner{inner}
6970     \renewcommand\newpage{\addtocounter{framenum}{1}}
6971     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6972     \renewenvironment{itemize}{
6973         \ifx\itemize@level\itemize@outer
6974             \def\itemize@label{\$ \rhd \$}
6975         \fi
6976         \ifx\itemize@level\itemize@inner
6977             \def\itemize@label{\$ \scriptstyle \rhd \$}
6978         \fi
6979         \begin{list}
6980             {\itemize@label}
6981             {\setlength{\labelsep}{.3em}
6982              \setlength{\labelwidth}{.5em}
6983              \setlength{\leftmargin}{1.5em}
6984             }

```

```

6985     \edef\itemize@level{\itemize@inner}
6986   }{
6987     \end{list}
6988   }

```

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

```

6989     \stex_html_backend:TF {
6990       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6991     }{
6992       \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6993     }
6994   }{
6995     \stex_html_backend:TF {
6996       \miko@slidelabel\egroup\end{stex_annotate_env}
6997     }\medskip\miko@slidelabel\end{mdframed}}
6998   }

```

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

`\frametitle`

```

6999   \renewcommand{\frametitle}[1]{
7000     \stex_document_title:n { #1 }
7001     {\Large\bf\sf\color{blue}{#1}}\medskip
7002   }
7003 }

```

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

EdN:19

`\pause`

```

19
7004 \bool_if:NT \c__notesslides_notes_bool {
7005   \newcommand\pause{}
7006 }

```

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

`nparagraph`

```

7007 \bool_if:NTF \c__notesslides_notes_bool {
7008   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]}\end{sparagraph}}
7009 }{
7010   \excludecomment{nparagraph}
7011 }

```

`nfragment`

```

7012 \bool_if:NTF \c__notesslides_notes_bool {
7013   \newenvironment{nfragment}[2][\begin{sfragment}[#1]{#2}]\end{sfragment}}
7014 }{
7015   \excludecomment{nfragment}
7016 }

```

`ndefinition`

```

7017 \bool_if:NTF \c__notesslides_notes_bool {
7018   \newenvironment{ndefinition}[1][\begin{sdefinition}[#1]}\end{sdefinition}}
7019 }{
7020   \excludecomment{ndefinition}
7021 }

```

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

nassertion

```

7022 \bool_if:NTF \c_notesslides_notes_bool {
7023   \newenvironment{nassertion}[1][\begin{sassertion}[#1]}\end{sassertion}}
7024 }{
7025   \excludecomment{nassertion}
7026 }

```

nsproof

```

7027 \bool_if:NTF \c_notesslides_notes_bool {
7028   \newenvironment{nproof}[2][\begin{sproof}[#1]{#2}]\end{sproof}}
7029 }{
7030   \excludecomment{nproof}
7031 }

```

nexample

```

7032 \bool_if:NTF \c_notesslides_notes_bool {
7033   \newenvironment{nexample}[1][\begin{sexample}[#1]}\end{sexample}}
7034 }{
7035   \excludecomment{nexample}
7036 }

```

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

```

7037 \def\inputref@preskip{\smallskip}
7038 \def\inputref@postskip{\medskip}

```

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

\inputref*

```

7039 \let\orig@inputref\inputref
7040 \def\inputref{\@ifstar\ninputref\orig@inputref}
7041 \newcommand\ninputref[2][\{
7042   \bool_if:NT \c_notesslides_notes_bool {
7043     \orig@inputref[#1]{#2}
7044   }
7045 }

```

(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 \TeX logo. Customization can be done by \setslideologo{<logo name>}.

```

7046 \newlength{\slideologoheight}
7047
7048 \bool_if:NTF \c_notesslides_notes_bool {
7049   \setlength{\slideologoheight}{.4cm}
7050 }{
7051   \setlength{\slideologoheight}{1cm}
7052 }
7053 \newsavebox{\slideologo}

```

```

7054 \sbox{\slidelogo}{\TeX}
7055 \newrobustcmd{\setslidelogo}[1]{
7056   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
7057 }

```

(End definition for `\setslidelogo`. 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.

```

7058 \def\source{Michael Kohlhase}% customize locally
7059 \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.

```

7060 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7061 \newsavebox{\cclogo}
7062 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
7063 \newif\ifcchref\cchreffalse
7064 \AtBeginDocument{
7065   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7066 }
7067 \def\licensing{
7068   \ifcchref
7069     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7070   \else
7071     {\usebox{\cclogo}}
7072   \fi
7073 }
7074 \newrobustcmd{\setlicensing}[2][{}]{
7075   \def\@url{#1}
7076   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
7077   \ifx\@url\@empty
7078     \def\licensing{{\usebox{\cclogo}}}
7079   \else
7080     \def\licensing{
7081       \ifcchref
7082         \href{#1}{\usebox{\cclogo}}
7083       \else
7084         {\usebox{\cclogo}}
7085       \fi
7086     }
7087   \fi
7088 }

```

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

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

```

7089 \newrobustcmd\miko@slidelabel{
7090   \vbox to \slidelogoheight{

```

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


```

7091 \vss\hbox to \slidewidth
7092 {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slide\logo}}
7093 }
7094 }

```

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

38.4 Frame Images

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

```

7095 \def\Gin@mhrepos{}
7096 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7097 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7098 \newrobustcmd\frameimage[2][]{
7099   \stepcounter{slide}
7100   \bool_if:NT \c__notesslides_frameimages_bool {
7101     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7102     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7103     \begin{center}
7104       \bool_if:NTF \c__notesslides_fiboxed_bool {
7105         \fbox{
7106           \ifx\Gin@ewidth\@empty
7107             \ifx\Gin@mhrepos\@empty
7108               \mhgraphics[width=\slidewidth,#1]{#2}
7109             \else
7110               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7111             \fi
7112           \else% Gin@ewidth empty
7113             \ifx\Gin@mhrepos\@empty
7114               \mhgraphics[#1]{#2}
7115             \else
7116               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7117             \fi
7118           \fi% Gin@ewidth empty
7119         }
7120       }{
7121         \ifx\Gin@ewidth\@empty
7122           \ifx\Gin@mhrepos\@empty
7123             \mhgraphics[width=\slidewidth,#1]{#2}
7124           \else
7125             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7126           \fi
7127           \ifx\Gin@mhrepos\@empty
7128             \mhgraphics[#1]{#2}
7129           \else
7130             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7131           \fi
7132         \fi% Gin@ewidth empty
7133       }
7134     \end{center}
7135     \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7136     \bool_if:NF \c__notesslides_notes_bool { \vfill }

```

```

7137 }
7138 } % 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.

```

7139 \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

```

7140 \AddToHook{begindocument}{
7141   \definecolor{green}{rgb}{0,.5,0}
7142   \definecolor{purple}{cmk}{.3,1,0,.17}
7143 }

```

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`.

```

7144 % \def\STpresent#1{\textcolor{blue}{#1}}
7145 \def\defemph#1{\textcolor{magenta}{#1}}
7146 \def\symrefemph#1{\textcolor{cyan}{#1}}
7147 \def\compemph#1{\textcolor{blue}{#1}}
7148 \def\titleemph#1{\textcolor{blue}{#1}}
7149 \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.

```

7150 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7151 \def\smalltextwarning{
7152   \pgfuseimage{miko@small@dbend}
7153   \xspace
7154 }
7155 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7156 \newrobustcmd\textwarning{
7157   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7158   \xspace
7159 }
7160 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7161 \newrobustcmd\bigtextwarning{
7162   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7163   \xspace
7164 }

```

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

```

7165 \newrobustcmd\putgraphicsat[3]{
7166   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7167 }
7168 \newrobustcmd\putat[2]{
7169   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7170 }

```

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`.

```

7171 \bool_if:NT \c__notesslides_sectocframes_bool {
7172   \str_if_eq:VnTF \__notesslidesstopsect{part}{
7173     \newcounter{chapter}\counterwithin*{section}{chapter}
7174   }{
7175     \str_if_eq:VnTF \__notesslidesstopsect{chapter}{
7176       \newcounter{chapter}\counterwithin*{section}{chapter}
7177     }
7178   }
7179 }
```

`\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

7180 \def\part@prefix{}
7181 \@ifpackageloaded{document-structure}{}{
7182   \str_case:VnF \__notesslidesstopsect {
7183     {part}{
7184       \int_set:Nn \l_document_structure_section_level_int {0}
7185       \def\thesection{\arabic{chapter}.\arabic{section}}
7186       \def\part@prefix{\arabic{chapter}.}
7187     }
7188     {chapter}{
7189       \int_set:Nn \l_document_structure_section_level_int {1}
7190       \def\thesection{\arabic{chapter}.\arabic{section}}
7191       \def\part@prefix{\arabic{chapter}.}
7192     }
7193   }{
7194     \int_set:Nn \l_document_structure_section_level_int {2}
7195     \def\part@prefix{}
7196   }
7197 }
7198
7199 \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

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

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

sfragment

```

7200 \renewenvironment{sfragment}[2][]{
7201   \__document_structure_sfragment_args:n { #1 }
7202   \int_incr:N \l_document_structure_section_level_int
7203   \bool_if:NT \c__notesslides_sectocframes_bool {
7204     \stepcounter{slide}
7205     \begin{frame}[noframenumbering]
7206     \vfill\Large\centering
7207     \red{
7208       \ifcase\l_document_structure_section_level_int\or
```

```

7209         \stepcounter{part}
7210         \def\__notesslideslabel{\o{omdoc@part@kw}~\Roman{part}}
7211         \def\currentsectionlevel{\o{omdoc@part@kw}}
7212     \or
7213         \stepcounter{chapter}
7214         \def\__notesslideslabel{\o{omdoc@chapter@kw}~\arabic{chapter}}
7215         \def\currentsectionlevel{\o{omdoc@chapter@kw}}
7216     \or
7217         \stepcounter{section}
7218         \def\__notesslideslabel{\part@prefix\arabic{section}}
7219         \def\currentsectionlevel{\o{omdoc@section@kw}}
7220     \or
7221         \stepcounter{subsection}
7222         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7223         \def\currentsectionlevel{\o{omdoc@subsection@kw}}
7224     \or
7225         \stepcounter{subsubsection}
7226         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7227         \def\currentsectionlevel{\o{omdoc@subsubsection@kw}}
7228     \or
7229         \stepcounter{paragraph}
7230         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7231         \def\currentsectionlevel{\o{omdoc@paragraph@kw}}
7232     \else
7233         \def\__notesslideslabel{}
7234         \def\currentsectionlevel{\o{omdoc@paragraph@kw}}
7235     \fi% end ifcase
7236     \__notesslideslabel%\sref@label@id\__notesslideslabel
7237     \quad #2%
7238 }%
7239 \vfill%
7240 \end{frame}%
7241 }
7242 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7243     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7244 }
7245 }{}
7246 }

```

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

```

7247 \def\inserttheorembodyfont{\normalfont}
7248 %\bool_if:NF \c__notesslides_notes_bool {
7249 % \defbeamertemplate{theorem begin}{miko}
7250 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7251 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7252 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7253 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7254 % \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.

```

7255 % \expandafter\def\csname Parent2\endcsname{}

```

```

7256 %}
7257
7258 \AddToHook{begindocument}{% this does not work for some reasons
7259   \setbeamertemplate{theorems}[ams style]
7260 }
7261 \bool_if:NT \c__notesslides_notes_bool {
7262   \renewenvironment{columns}[1][]{%
7263     \par\noindent%
7264     \begin{minipage}%
7265       \slidewidth\centering\leavevmode%
7266   }{%
7267     \end{minipage}\par\noindent%
7268   }%
7269   \newsavebox\columnbox%
7270   \renewenvironment<>{column}[2][]{%
7271     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7272   }{%
7273     \end{minipage}\end{lrbox}\usebox\columnbox%
7274   }%
7275 }
7276 \bool_if:NTF \c__notesslides_noproblems_bool {
7277   \newenvironment{problems}{}{}
7278 }{
7279   \excludcomment{problems}
7280 }

```

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.

```

7281 \gdef\printexcursions{}
7282 \newcommand\excursionref[2]{% label, text
7283   \bool_if:NT \c__notesslides_notes_bool {
7284     \begin{sparagraph}[title=Excursion]
7285       #2 \sref[fallback=the appendix]{#1}.
7286     \end{sparagraph}
7287   }
7288 }
7289 \newcommand\activate@excursion[2][{}{
7290   \gappto\printexcursions{\inputref[#1]{#2}}
7291 }
7292 \newcommand\excursion[4][{}{repos, label, path, text
7293   \bool_if:NT \c__notesslides_notes_bool {
7294     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7295   }
7296 }

```

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

\excursiongroup

```

7297 \keys_define:nn{notesslides / excursiongroup }{

```

```

7298   id      .str_set_x:N = \l__notesslides_excursion_id_str,
7299   intro   .tl_set:N    = \l__notesslides_excursion_intro_tl,
7300   mhrepos .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7301 }
7302 \cs_new_protected:Nn \l__notesslides_excursion_args:n {
7303   \tl_clear:N \l__notesslides_excursion_intro_tl
7304   \str_clear:N \l__notesslides_excursion_id_str
7305   \str_clear:N \l__notesslides_excursion_mhrepos_str
7306   \keys_set:nn {notesslides / excursiongroup} { #1 }
7307 }
7308 \newcommand\excursiongroup[1][ ]{
7309   \l__notesslides_excursion_args:n{ #1 }
7310   \ifdefempty\printexcursions{}% only if there are excursions
7311   {\begin{note}
7312     \begin{sfragment}[#1]{Excursions}%
7313     \ifdefempty\l__notesslides_excursion_intro_tl}{
7314       \inputref[\l__notesslides_excursion_mhrepos_str]{
7315         \l__notesslides_excursion_intro_tl
7316       }
7317     }
7318     \printexcursions%
7319   \end{sfragment}
7320   \end{note}}
7321 }
7322 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7323 \</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.

```
7324 <*package>
7325 <@@=problems>
7326 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7327 \RequirePackage{l3keys2e,stex}
7328
7329 \keys_define:nn { problem / pkg }{
7330   notes      .default:n    = { true },
7331   notes      .bool_set:N   = \c__problems_notes_bool,
7332   gnotes     .default:n    = { true },
7333   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7334   hints      .default:n    = { true },
7335   hints      .bool_set:N   = \c__problems_hints_bool,
7336   solutions  .default:n    = { true },
7337   solutions  .bool_set:N   = \c__problems_solutions_bool,
7338   pts        .default:n    = { true },
7339   pts        .bool_set:N   = \c__problems_pts_bool,
7340   min        .default:n    = { true },
7341   min        .bool_set:N   = \c__problems_min_bool,
7342   boxed      .default:n    = { true },
7343   boxed      .bool_set:N   = \c__problems_boxed_bool,
7344   unknown    .code:n       = {}
7345 }
7346 \newif\ifsolutions
7347
7348 \ProcessKeysOptions{ problem / pkg }
7349 \bool_if:NTF \c__problems_solutions_bool {
7350   \solutionstrue
7351 }{
7352   \solutionsfalse
7353 }
```

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

```
7354 \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.

```
7355 \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.

```
7356 \def\prob@problem@kw{Problem}
7357 \def\prob@solution@kw{Solution}
7358 \def\prob@hint@kw{Hint}
7359 \def\prob@note@kw{Note}
7360 \def\prob@gnote@kw{Grading}
7361 \def\prob@pt@kw{pt}
7362 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7363 \AddToHook{begindocument}{
7364   \ltx@ifpackageloaded{babel}{
7365     \makeatletter
7366     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7367     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7368       \input{problem-ngerman.ldf}
7369     }
7370     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7371       \input{problem-finnish.ldf}
7372     }
7373     \clist_if_in:NnT \l_tmpa_clist {french}{
7374       \input{problem-french.ldf}
7375     }
7376     \clist_if_in:NnT \l_tmpa_clist {russian}{
7377       \input{problem-russian.ldf}
7378     }
7379     \makeatother
7380   }{}
7381 }
```

39.2 Problems and Solutions

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

```
7382 \keys_define:nn{ problem / problem }{
7383   id      .str_set_x:N = \l__problems_prob_id_str,
7384   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7385   min     .tl_set:N    = \l__problems_prob_min_tl,
7386   title   .tl_set:N    = \l__problems_prob_title_tl,
7387   type    .tl_set:N    = \l__problems_prob_type_tl,
7388   imports .tl_set:N    = \l__problems_prob_imports_tl,
7389   name    .str_set_x:N = \l__problems_prob_name_str,
7390   refnum  .int_set:N   = \l__problems_prob_refnum_int
```



```

7391 }
7392 \cs_new_protected:Nn \__problems_prob_args:n {
7393   \str_clear:N \l__problems_prob_id_str
7394   \str_clear:N \l__problems_prob_name_str
7395   \tl_clear:N \l__problems_prob_pts_tl
7396   \tl_clear:N \l__problems_prob_min_tl
7397   \tl_clear:N \l__problems_prob_title_tl
7398   \tl_clear:N \l__problems_prob_type_tl
7399   \tl_clear:N \l__problems_prob_imports_tl
7400   \int_zero_new:N \l__problems_prob_refnum_int
7401   \keys_set:nn { problem / problem }{ #1 }
7402   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7403     \let\l__problems_prob_refnum_int\undefined
7404   }
7405 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7406 \newcounter{problem}[section]
7407 \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.

```

7408 \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

```

7409 \newcommand\prob@number{
7410   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7411     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7412   }{
7413     \int_if_exist:NTF \l__problems_prob_refnum_int {
7414       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7415     }{
7416       \prob@label\theproblem
7417     }
7418   }
7419 }

```

(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.

```

7420 \newcommand\prob@title[3]{%
7421   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7422     #2 \l__problems_inclprob_title_tl #3
7423   }{
7424     \tl_if_exist:NTF \l__problems_prob_title_tl {
7425       #2 \l__problems_prob_title_tl #3
7426     }{
7427       #1

```

```

7428     }
7429   }
7430 }

```

(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.

```

7431 \def\prob@heading{
7432   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7433   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7434 }

```

(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`

```

7435 \newenvironment{sproblem}[1][{}]{
7436   \__problems_prob_args:n{#1}%\sref@target%
7437   \@in@omtexttrue% we are in a statement (for inline definitions)
7438   \stepcounter{problem}\record@problem
7439   \def\current@section@level{\prob@problem@kw}
7440
7441   \str_if_empty:NT \l__problems_prob_name_str {
7442     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7443     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7444     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7445   }
7446   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7447
7448   \stex_reactivate_macro:N \STEXexport
7449   \stex_reactivate_macro:N \importmodule
7450   \stex_reactivate_macro:N \symdecl
7451   \stex_reactivate_macro:N \notation
7452   \stex_reactivate_macro:N \symdef
7453
7454   \stex_if_do_html:T{
7455     \begin{stex_annotate_env} {problem} {
7456       \l_stex_module_ns_str ? \l_stex_module_name_str
7457     }
7458
7459     \stex_annotate_invisible:nnn{header}{~}{~} {
7460       \stex_annotate:nnn{language}{~}{~} \l_stex_module_lang_str }{~}
7461       \stex_annotate:nnn{signature}{~}{~} \l_stex_module_sig_str }{~}
7462       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7463         \stex_annotate:nnn{metatheory}{~}{~} \l_stex_module_meta_str }{~}
7464     }
7465   }
7466 }
7467

```

```

7468 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7469
7470
7471 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7472   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7473 }{
7474   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7475 }
7476 \str_if_exist:NTF \l__problems_inclprob_id_str {
7477   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7478 }{
7479   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7480 }
7481
7482
7483 \stex_if_smsmode:F {
7484   \clist_set:No \l_tmpa_clist \sproblemtype
7485   \tl_clear:N \l_tmpa_tl
7486   \clist_map_inline:Nn \l_tmpa_clist {
7487     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7488       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7489     }
7490   }
7491   \tl_if_empty:NTF \l_tmpa_tl {
7492     \__problems_sproblem_start:
7493   }{
7494     \l_tmpa_tl
7495   }
7496 }
7497 \stex_ref_new_doc_target:n \sproblemid
7498 \stex_smsmode_do:
7499 }{
7500   \__stex_modules_end_module:
7501   \stex_if_smsmode:F{
7502     \clist_set:No \l_tmpa_clist \sproblemtype
7503     \tl_clear:N \l_tmpa_tl
7504     \clist_map_inline:Nn \l_tmpa_clist {
7505       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7506         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7507       }
7508     }
7509     \tl_if_empty:NTF \l_tmpa_tl {
7510       \__problems_sproblem_end:
7511     }{
7512       \l_tmpa_tl
7513     }
7514   }
7515   \stex_if_do_html:T{
7516     \end{stex_annotate_env}
7517   }
7518
7519   \smallskip
7520 }
7521

```

```

7522 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7523
7524
7525
7526 \cs_new_protected:Nn \__problems_sproblem_start: {
7527   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7528 }
7529 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7530
7531 \newcommand\stexpatchproblem[3][] {
7532   \str_set:Nx \l_tmpa_str{ #1 }
7533   \str_if_empty:NTF \l_tmpa_str {
7534     \tl_set:Nn \__problems_sproblem_start: { #2 }
7535     \tl_set:Nn \__problems_sproblem_end: { #3 }
7536   }{
7537     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7538     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7539   }
7540 }
7541
7542
7543 \bool_if:NT \c__problems_boxed_bool {
7544   \surroundwithhmdframed{problem}
7545 }

```

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

```

7546 \def\record@problem{
7547   \protected@write\@auxout{}
7548   {
7549     \string\@problem{\prob@number}
7550     {
7551       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7552         \l__problems_inclprob_pts_tl
7553       }{
7554         \l__problems_prob_pts_tl
7555       }
7556     }%
7557     {
7558       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7559         \l__problems_inclprob_min_tl
7560       }{
7561         \l__problems_prob_min_tl
7562       }
7563     }
7564   }
7565 }

```

(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).

```

7566 \def\@problem#1#2#3{}

```

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

solution The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has its own keys that we need to define first.

```

7567 \keys_define:nn { problem / solution }{
7568   id          .str_set_x:N = \l__problems_solution_id_str ,
7569   for         .tl_set:N    = \l__problems_solution_for_tl ,
7570   height      .dim_set:N   = \l__problems_solution_height_dim ,
7571   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7572   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7573   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7574 }
7575 \cs_new_protected:Nn \__problems_solution_args:n {
7576   \str_clear:N \l__problems_solution_id_str
7577   \tl_clear:N \l__problems_solution_for_tl
7578   \tl_clear:N \l__problems_solution_srccite_tl
7579   \clist_clear:N \l__problems_solution_creators_clist
7580   \clist_clear:N \l__problems_solution_contributors_clist
7581   \dim_zero:N \l__problems_solution_height_dim
7582   \keys_set:nn { problem / solution }{ #1 }
7583 }

```

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

```

7584 \newcommand\@startsolution[1][ ]{
7585   \__problems_solution_args:n { #1 }
7586   \@in@omtexttrue% we are in a statement.
7587   \bool_if:NF \c__problems_boxed_bool { \hrule }
7588   \smallskip\noindent
7589   {\textbf\prob@solution@kw : \enspace}
7590   \begin{small}
7591   \def\current@section@level{\prob@solution@kw}
7592   \ignorespacesandpars
7593 }

```

\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.

```

7594 \box_new:N \l__problems_solution_box
7595 \newenvironment{solution}[1][ ]{
7596   \stex_html_backend:TF{
7597     \stex_if_do_html:T{
7598       \begin{stex_annotate_env}{solution}{ }
7599     }
7600   }{
7601     \setbox\l__problems_solution_box\vbox\bgroup
7602     \par\smallskip\hrule\smallskip
7603     \noindent\textbf{Solution:}~
7604   }
7605 }{
7606   \stex_html_backend:TF{
7607     \stex_if_do_html:T{
7608       \end{stex_annotate_env}
7609     }
7610   }{

```

```

7611 \smallskip\hrule
7612 \egroup
7613 \bool_if:NT \c__problems_solutions_bool {
7614 \box\l__problems_solution_box
7615 }
7616 }
7617 }
7618
7619 \newcommand\startsolutions{
7620 \bool_set_true:N \c__problems_solutions_bool
7621 % \specialcomment{solution}{\@startsolution}{
7622 % \bool_if:NF \c__problems_boxed_bool {
7623 % \hrule\medskip
7624 % }
7625 % \end{small}%
7626 % }
7627 % \bool_if:NT \c__problems_boxed_bool {
7628 % \surroundwithmdframed{solution}
7629 % }
7630 }

```

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

\stopsolutions

```

7631 \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.

```

7632 \ifsolutions
7633 \startsolutions
7634 \else
7635 \stopsolutions
7636 \fi

```

exnote

```

7637 \bool_if:NTF \c__problems_notes_bool {
7638 \newenvironment{exnote}[1][]{
7639 \par\smallskip\hrule\smallskip
7640 \noindent\textbf{\prob@note@kw :~ }\small
7641 }{
7642 \smallskip\hrule
7643 }
7644 }{
7645 \excludecomment{exnote}
7646 }

```

hint

```

7647 \bool_if:NTF \c__problems_notes_bool {
7648 \newenvironment{hint}[1][]{
7649 \par\smallskip\hrule\smallskip
7650 \noindent\textbf{\prob@hint@kw :~ }\small
7651 }{
7652 \smallskip\hrule
7653 }

```

```

7654 \newenvironment{exhint}[1][]{
7655   \par\smallskip\hrule\smallskip
7656   \noindent\textbf{\prob@hint@kw :~ }\small
7657 }{
7658   \smallskip\hrule
7659 }
7660 }{
7661   \excludecomment{hint}
7662   \excludecomment{exhint}
7663 }

```

gnote

```

7664 \bool_if:NTF \c__problems_notes_bool {
7665   \newenvironment{gnote}[1][]{
7666     \par\smallskip\hrule\smallskip
7667     \noindent\textbf{\prob@gnote@kw :~ }\small
7668   }{
7669     \smallskip\hrule
7670   }
7671   }{
7672     \excludecomment{gnote}
7673   }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7674 \newenvironment{mcb}{
7675   \begin{enumerate}
7676 }{
7677   \end{enumerate}
7678 }

```

we define the keys for the mcb macro

```

7679 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7680   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7681     \bool_set_true:N #1
7682   }{
7683     \bool_set_false:N #1
7684   }
7685 }
7686 \keys_define:nn { problem / mcb }{
7687   id .str_set:N = \l__problems_mcc_id_str ,
7688   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7689   T .default:n = { false } ,
7690   T .bool_set:N = \l__problems_mcc_t_bool ,
7691   F .default:n = { false } ,
7692   F .bool_set:N = \l__problems_mcc_f_bool ,
7693   Ttext .tl_set:N = \l__problems_mcc_Ttext_str ,
7694   Ftext .tl_set:N = \l__problems_mcc_Ftext_str
7695 }
7696 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

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

```

7697 \str_clear:N \l__problems_mcc_id_str
7698 \tl_clear:N \l__problems_mcc_feedback_tl
7699 \bool_set_false:N \l__problems_mcc_t_bool
7700 \bool_set_false:N \l__problems_mcc_f_bool
7701 \tl_clear:N \l__problems_mcc_Ttext_tl
7702 \tl_clear:N \l__problems_mcc_Ftext_tl
7703 \str_clear:N \l__problems_mcc_id_str
7704 \keys_set:nn { problem / mcc }{ #1 }
7705 }

```

\mcc

```

7706 \def\mccTrueText{\textbf{(true)~}}
7707 \def\mccFalseText{\textbf{(false)~}}
7708 \newcommand\mcc[2][] {
7709   \l__problems_mcc_args:n{ #1 }
7710   \item[$\Box$] #2
7711   \ifsolutions
7712     \l
7713     \bool_if:NT \l__problems_mcc_t_bool {
7714       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7715     }
7716     \bool_if:NT \l__problems_mcc_f_bool {
7717       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7718     }
7719     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7720       \emph{(\l__problems_mcc_feedback_tl)}
7721     }
7722   \fi
7723 } %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.

```

7724
7725 \keys_define:nn{ problem / inclproblem }{
7726   id      .str_set_x:N = \l__problems_inclprob_id_str,
7727   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7728   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7729   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7730   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7731   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7732   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7733 }
7734 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7735   \str_clear:N \l__problems_prob_id_str
7736   \tl_clear:N \l__problems_inclprob_pts_tl
7737   \tl_clear:N \l__problems_inclprob_min_tl
7738   \tl_clear:N \l__problems_inclprob_title_tl
7739   \tl_clear:N \l__problems_inclprob_type_tl

```



```

7740 \int_zero_new:N \l__problems_inclprob_refnum_int
7741 \str_clear:N \l__problems_inclprob_mhrepos_str
7742 \keys_set:nn { problem / inclproblem }{ #1 }
7743 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7744   \let\l__problems_inclprob_pts_tl\undefined
7745 }
7746 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7747   \let\l__problems_inclprob_min_tl\undefined
7748 }
7749 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7750   \let\l__problems_inclprob_title_tl\undefined
7751 }
7752 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7753   \let\l__problems_inclprob_type_tl\undefined
7754 }
7755 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7756   \let\l__problems_inclprob_refnum_int\undefined
7757 }
7758 }
7759
7760 \cs_new_protected:Nn \__problems_inclprob_clear: {
7761   \let\l__problems_inclprob_id_str\undefined
7762   \let\l__problems_inclprob_pts_tl\undefined
7763   \let\l__problems_inclprob_min_tl\undefined
7764   \let\l__problems_inclprob_title_tl\undefined
7765   \let\l__problems_inclprob_type_tl\undefined
7766   \let\l__problems_inclprob_refnum_int\undefined
7767   \let\l__problems_inclprob_mhrepos_str\undefined
7768 }
7769 \__problems_inclprob_clear:
7770
7771 \newcommand\includeproblem[2][ ]{
7772   \__problems_inclprob_args:n{ #1 }
7773   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7774     \stex_html_backend:TF {
7775       \str_clear:N \l_tmpa_str
7776       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7777         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7778       }
7779       \stex_annotate_invisible:nnn{includeproblem}{
7780         \l_tmpa_str / #2
7781       }{}
7782     }{
7783       \begingroup
7784         \inputreftrue
7785         \tl_if_empty:nTF{ ##1 }{
7786           \input{#2}
7787         }{
7788           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7789         }
7790       \endgroup
7791     }
7792   }
7793   \__problems_inclprob_clear:

```

```
7794 }
```

(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.

```
7795 \AddToHook{enddocument}{
7796   \bool_if:NT \c__problems_pts_bool {
7797     \message{Total:~\arabic{pts}~points}
7798   }
7799   \bool_if:NT \c__problems_min_bool {
7800     \message{Total:~\arabic{min}~minutes}
7801   }
7802 }
```

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

```
7803 \def\pts#1{
7804   \bool_if:NT \c__problems_pts_bool {
7805     \marginpar{#1~\prob@pt@kw}
7806   }
7807 }
7808 \def\min#1{
7809   \bool_if:NT \c__problems_min_bool {
7810     \marginpar{#1~\prob@min@kw}
7811   }
7812 }
```

`\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.

```
7813 \newcounter{pts}
7814 \def\show@pts{
7815   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7816     \bool_if:NT \c__problems_pts_bool {
7817       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7818       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7819     }
7820   }{
7821     \tl_if_exist:NT \l__problems_prob_pts_tl {
7822       \bool_if:NT \c__problems_pts_bool {
7823         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7824           \tl_set:Nn \l__problems_prob_pts_tl {0}
7825         }
7826         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7827         \addtocounter{pts}{\l__problems_prob_pts_tl}
7828       }
7829     }
7830   }
7831 }
```

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

\show@min

```

7832 \newcounter{min}
7833 \def\show@min{
7834   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7835     \bool_if:NT \c__problems_min_bool {
7836       \marginpar{\l__problems_inclprob_pts_tl\ min}
7837       \addtocounter{min}{\l__problems_inclprob_min_tl}
7838     }
7839   }{
7840     \tl_if_exist:NT \l__problems_prob_min_tl {
7841       \bool_if:NT \c__problems_min_bool {
7842         \tl_if_empty:NT\l__problems_prob_min_tl{
7843           \tl_set:Nn \l__problems_prob_min_tl {0}
7844         }
7845         \marginpar{\l__problems_prob_min_tl\ min}
7846         \addtocounter{min}{\l__problems_prob_min_tl}
7847       }
7848     }
7849   }
7850 }
7851 \</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.

```
7852 \*package>
7853 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7854 \RequirePackage{13keys2e}
7855
7856 \newif\iftest\testfalse
7857 \DeclareOption{test}{\testtrue}
7858 \newif\ifmultiple\multiplefalse
7859 \DeclareOption{multiple}{\multipletrue}
7860 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7861 \ProcessOptions
```

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

```
7862 \RequirePackage{keyval}[1997/11/10]
7863 \RequirePackage{problem}
```

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

```
7864 \newcommand\hwexam@assignment@kw{Assignment}
7865 \newcommand\hwexam@given@kw{Given}
7866 \newcommand\hwexam@due@kw{Due}
7867 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7868 blank~for~extra~space}
7869 \def\hwexam@minutes@kw{minutes}
7870 \newcommand\correction@probs@kw{prob.}
7871 \newcommand\correction@pts@kw{total}
7872 \newcommand\correction@reached@kw{reached}
7873 \newcommand\correction@sum@kw{Sum}
7874 \newcommand\correction@grade@kw{grade}
7875 \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

```

7876 \AddToHook{begindocument}{
7877 \ltx@ifpackageloaded{babel}{
7878 \makeatletter
7879 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7880 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7881 \input{hwexam-ngerman.ldf}
7882 }
7883 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7884 \input{hwexam-finnish.ldf}
7885 }
7886 \clist_if_in:NnT \l_tmpa_clist {french}{
7887 \input{hwexam-french.ldf}
7888 }
7889 \clist_if_in:NnT \l_tmpa_clist {russian}{
7890 \input{hwexam-russian.ldf}
7891 }
7892 \makeatother
7893 }{}
7894 }
7895

```

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.

```

7896 \newcounter{assignment}
7897 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7908 \keys_define:nn { hwexam / assignment } {
7909 id .str_set:N = \l_@@_assign_id_str,
7910 number .int_set:N = \l_@@_assign_number_int,
7911 title .tl_set:N = \l_@@_assign_title_tl,
7912 type .tl_set:N = \l_@@_assign_type_tl,
7913 given .tl_set:N = \l_@@_assign_given_tl,
7914 due .tl_set:N = \l_@@_assign_due_tl,
7915 loadmodules .code:n = {
7916 \bool_set_true:N \l_@@_assign_loadmodules_bool
7917 }
7918 }

7909 \cs_new_protected:Nn \_@@_assignment_args:n {
7910 \str_clear:N \l_@@_assign_id_str
7911 \int_set:Nn \l_@@_assign_number_int {-1}
7912 \tl_clear:N \l_@@_assign_title_tl
7913 \tl_clear:N \l_@@_assign_type_tl
7914 \tl_clear:N \l_@@_assign_given_tl
7915 \tl_clear:N \l_@@_assign_due_tl
7916 \bool_set_false:N \l_@@_assign_loadmodules_bool
7917 \keys_set:nn { hwexam / assignment }{ #1 }
7918 }

```

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.

```

7919 \newcommand\given@due[2]{
7920 \bool_lazy_all:nF {
7921 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7922 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7923 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7924 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7925 }{ #1 }
7926
7927 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7928 \tl_if_empty:NF \l_@@_assign_given_tl {
7929 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7930 }
7931 }{
7932 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7933 }
7934
7935 \bool_lazy_or:nnF {
7936 \bool_lazy_and_p:nn {
7937 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7938 }{
7939 \tl_if_empty_p:V \l_@@_assign_due_tl
7940 }
7941 }{
7942 \bool_lazy_and_p:nn {
7943 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7944 }{
7945 \tl_if_empty_p:V \l_@@_assign_due_tl
7946 }
7947 }{ ,~ }
7948
7949 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7950 \tl_if_empty:NF \l_@@_assign_due_tl {
7951 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7952 }
7953 }{
7954 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7955 }
7956
7957 \bool_lazy_all:nF {
7958 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7959 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7960 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7961 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7962 }{ #2 }
7963 }

```

`\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.

```

7964 \newcommand\assignment@title[3]{
7965 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7966 \tl_if_empty:NTF \l_@@_assign_title_tl {
7967 #1
7968 }{
7969 #2\l_@@_assign_title_tl#3
7970 }
7971 }{
7972 #2\l_@@_inclasssign_title_tl#3
7973 }
7974 }

```

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

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

```

7975 \newcommand\assignment@number{
7976 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
7977 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7978 \arabic{assignment}
7979 } {
7980 \int_use:N \l_@@_assign_number_int
7981 }
7982 }{
7983 \int_use:N \l_@@_inclasssign_number_int
7984 }
7985 }

```

(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.

```

7986 \newenvironment{assignment}[1][]{
7987 \_@@_assignment_args:n { #1 }
7988 %\sref@target
7989 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7990 \global\stepcounter{assignment}
7991 }{
7992 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7993 }
7994 \setcounter{problem}{0}
7995 \renewcommand\prob@label[1]{\assignment@number.##1}
7996 \def\current@section@level{\document@hwexamtype}
7997 %\sref@label{id}{\document@hwexamtype \thesection}
7998 \begin{@assignment}
7999 }{
8000 \end{@assignment}
8001 }

```

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

```

8002 \def\ass@title{
8003 {\protect\document@hwexamtype}\arabic{assignment}
8004 \assignment@title{}\;{}{}\;} -- \given@due{}\}
8005 }
8006 \ifmultiple
8007 \newenvironment{@assignment}{
8008 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8009 \begin{sfragment}[loadmodules]{\ass@title}
8010 }{
8011 \begin{sfragment}{\ass@title}
8012 }
8013 }{
8014 \end{sfragment}
8015 }

```

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

```

8016 \else
8017 \newenvironment{@assignment}{
8018 \begin{center}\bf
8019 \Large@title\strut\
8020 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;{}{}\;{}\\}
8021 \large\given@due{--\;}\;{};--}
8022 \end{center}
8023 }{}
8024 \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.

```

8025 \keys_define:nn { hwexam / inclassignment } {
8026 %id .str_set_x:N = \l_@@_assign_id_str,
8027 number .int_set:N = \l_@@_inclassign_number_int,
8028 title .tl_set:N = \l_@@_inclassign_title_tl,
8029 type .tl_set:N = \l_@@_inclassign_type_tl,
8030 given .tl_set:N = \l_@@_inclassign_given_tl,
8031 due .tl_set:N = \l_@@_inclassign_due_tl,
8032 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8033 }
8034 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8035 \int_set:Nn \l_@@_inclassign_number_int {-1}
8036 \tl_clear:N \l_@@_inclassign_title_tl
8037 \tl_clear:N \l_@@_inclassign_type_tl
8038 \tl_clear:N \l_@@_inclassign_given_tl
8039 \tl_clear:N \l_@@_inclassign_due_tl
8040 \str_clear:N \l_@@_inclassign_mhrepos_str
8041 \keys_set:nn { hwexam / inclassignment }{ #1 }
8042 }
8043 \l_@@_inclassignment_args:n {}
8044
8045 \newcommand\inputassignment[2][{}]{

```



```

8046 \_@@_inclassassignment_args:n { #1 }
8047 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8048 \input{#2}
8049 }{
8050 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8051 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8052 }
8053 }
8054 \_@@_inclassassignment_args:n {}
8055 }
8056 \newcommand\includeassignment[2][]{
8057 \newpage
8058 \inputassignment[#1]{#2}
8059 }

```

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

40.4 Typesetting Exams

\quizheading

```

8060 \ExplSyntaxOff
8061 \newcommand\quizheading[1]{%
8062 \def\@tas{#1}%
8063 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8064 \ifx\@tas\@empty\else%
8065 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8066 \fi%
8067 }
8068 \ExplSyntaxOn

```

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

\testheading

```

8069
8070 \def\hwexamheader{\input{hwexam-default.header}}
8071
8072 \def\hwexamminutes{
8073 \tl_if_empty:NTF \testheading@duration {
8074 {\testheading@min}~\hwexam@minutes@kw
8075 }{
8076 \testheading@duration
8077 }
8078 }
8079
8080 \keys_define:nn { hwexam / testheading } {
8081 min .tl_set:N = \testheading@min,
8082 duration .tl_set:N = \testheading@duration,
8083 reqpts .tl_set:N = \testheading@reqpts,
8084 tools .tl_set:N = \testheading@tools
8085 }
8086 \cs_new_protected:Nn \_@@_testheading_args:n {
8087 \tl_clear:N \testheading@min
8088 \tl_clear:N \testheading@duration

```

```

8089 \tl_clear:N \testheading@reqpts
8090 \tl_clear:N \testheading@tools
8091 \keys_set:nn { hwexam / testheading }{ #1 }
8092 }
8093 \newenvironment{testheading}[1][]{
8094 \_@@_testheading_args:n{ #1 }
8095 \newcount\check@time\check@time=\testheading@min
8096 \advance\check@time by -\theassignment@totalmin
8097 \newif\if@bonuspoints
8098 \tl_if_empty:NTF \testheading@reqpts {
8099 \@bonuspointsfalse
8100 }{
8101 \newcount\bonus@pts
8102 \bonus@pts=\theassignment@totalpts
8103 \advance\bonus@pts by -\testheading@reqpts
8104 \edef\bonus@pts{\the\bonus@pts}
8105 \@bonuspointstrue
8106 }
8107 \edef\check@time{\the\check@time}
8108
8109 \makeatletter\hwexamheader\makeatother
8110 }{
8111 \newpage
8112 }

```

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

\testspace

```

8113 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8114 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8115 \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.

```

8116 <@@=problems>
8117 \renewcommand\@problem[3]{
8118 \stepcounter{assignment@probs}
8119 \def\__problemspts{#2}
8120 \ifx\__problemspts\@empty\else
8121 \addtocounter{assignment@totalpts}{#2}
8122 \fi
8123 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8124 \xdef\correction@probs{\correction@probs & #1}%
8125 \xdef\correction@pts{\correction@pts & #2}
8126 \xdef\correction@reached{\correction@reached &}

```

```

8127 }
8128 <@@=hwexam>

```

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

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

```

8129 \newcounter{assignment@probs}
8130 \newcounter{assignment@totalpts}
8131 \newcounter{assignment@totalmin}
8132 \def\correction@probs{\correction@probs@kw}
8133 \def\correction@pts{\correction@pts@kw}
8134 \def\correction@reached{\correction@reached@kw}
8135 \stepcounter{assignment@probs}
8136 \newcommand\correction@table{
8137 \resizebox{\textwidth}{!}{%
8138 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8139 &\multicolumn{\theassignment@probs}{c|}|%|
8140 {\footnotesize\correction@forgrading@kw} &\\\hline
8141 \correction@probs & \correction@sum@kw & \correction@grade@kw\\\hline
8142 \correction@pts & \theassignment@totalpts & \\\hline
8143 \correction@reached & & \[.7cm]\hline
8144 \end{tabular}}
8145 </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/graphics.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

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).