

The sTeX3 Package Collection *

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

2022-04-27

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 2022-04-27)

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 \LaTeX concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

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

1. way of writing $\text{\texttt{L\text{A}TeX}}$ more modularly (object-oriented Math) for creating PDF documents or
2. foundation for authoring active documents in HTML5 instrumented with knowledge management services.

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

2.1.1 Minimal Setup for the PDF-only Workflow

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

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

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

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

```
<OMBIND>
  <OMID name="...?series?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 $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ 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 $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ archive:

- `/source/mod/` – individual $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ 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 $\text{\texttt{S}}\text{\texttt{T}}\text{\texttt{E}}\text{\texttt{X}}$ (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

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

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

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

`source-base` or

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

```

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

```

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

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

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

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

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

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

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

Output:

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

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

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

So to allow modular specification and facilitate re-use of document fragments \TeX 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.
- \rightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \rightarrow OMDOC/MMT – being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

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

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

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

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

In the example above:



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

3.3.6 Variables

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

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

`\svar`

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

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

\vardef

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

Example 19

Input:

```

1 \vardef{varf}[
2   name=f,
3   type=\funtype{\Nat}{\Nat},
4   op=f,
5   args=1,
6   prec=0;\neginfp
7 ]{\comp{f}#1}
8 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
9 \vardef{varx}[name=x,type=\Nat]{\comp{x}}
10
11 Given a function $\varf!:\funtype{\Nat}{\Nat}$,
12 by $\addition{\varf!,\varn}$ we mean the function
13 $\fun{\varx}{\varf{\addition{\varx,\varn}}}$

```

Output:

Given a function $f : \mathbb{N} \rightarrow \mathbb{N}$, by $f+n$ we mean the function $x \mapsto f(x+n)$

(of course, “lifting” addition in the way described in the previous example is an operation that deserves its own symbol rather than abusing `\addition`, but... well.)

TODO: bind=forall/exists

3.3.7 Variable Sequences

Variable *sequences* occur quite frequently in informal mathematics, hence they deserve special support. Variable sequences behave like variables in that they disappear at the end of the current \TeX group and are not exported from modules, but their declaration is quite different.

\varseq

A variable sequence is introduced via the command `\varseq`, which takes the usual optional arguments `name` and `type`. It then takes a starting index, an end index and a *notation* for the individual elements of the sequence parametric in an index. Note that both the starting as well as the ending index may be variables.

This is best shown by example:

Example 20

Input:


```

1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
3
4 The  $i$ th index of  $\seqa!$  is  $\seqa{i}$ .

```

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

```

Output:

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

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

Example 23

Input:

```

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

```

Output:

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

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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



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



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

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

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

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

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

`\STEXexport`

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



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

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



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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

A `monoid` is...

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

Example 25

Input:

```

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

```

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

\instantiate

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

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

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

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

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

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

`\varinstantiate`

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

This allows us to do things like:

Example 27

Input:

```

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

```

Output:

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

.

and

Example 28

Input:

```

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

```

Output:

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

.

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

.

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

Example 30

Input:

```

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



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

\arg

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

Output:

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

adding k yields...

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

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

Example 37

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

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

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 38

Input:

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

Output:

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

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

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

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

`\definiens`

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

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

Example 39

Input:

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

```

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

```

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

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

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

.

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

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

```

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

```

⁸EDNOTE: MK: reference

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

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

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

```

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

```

This yields the following result:

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

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

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

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

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

1.3. $n > 1$:

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

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

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

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

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

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

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

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

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

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

<hr/> <hr/> <code>\spfjust</code>	This evidence is marked up with the <code>\spfjust</code> macro in the <code>stex-proofs</code> package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional <code>KeyVal</code> argument, which can have the <code>method</code> key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).
<hr/> <hr/> <code>\premise</code>	The <code>\premise</code> macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the running example we have used the <code>\premise</code> macro to identify the inductive hypothesis.
<hr/> <hr/> <code>\justarg</code>	<p>The <code>\justarg</code> macro is very similar to <code>\premise</code> with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of <code>\premise</code>. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a <code>\justarg</code> macro.</p> <p>Note that both <code>\premise</code> and <code>\justarg</code> can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.</p>
<code>subproof</code>	The <code>spfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
<code>spfcases</code>	The <code>spfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcases</code> environments that mark up the cases one by one.
<code>spfcases</code>	The content of a <code>spfcases</code> environment are a sequence of case proofs marked up in the <code>spfcases</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>spfcases</code> environment is the same as that of a <code>proof</code> , i.e. <code>spfsteps</code> , <code>spfcomments</code> , and <code>spfcases</code> environments.
<hr/> <hr/> <code>\spfcasesketch</code>	<code>\spfcasesketch</code> is a variant of the <code>spfcases</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>spfcomment</code>	The <code>spfcomment</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>}{<ctext>}` tests the content of the global variable `<vname>`, only if (after expansion) it is equal to `<val>`, the conditional text `<ctext>` is formatted.

7.3 Slides and Course Notes

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

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

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

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



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

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

```

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

```

`\ifnotes`

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

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



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



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

`\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

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

`\setlicensing`

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

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

`\frameimage`
`\mhframeimage`

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

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

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

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

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

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

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

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

`\textwarning`

The `\textwarning` macro generates a warning sign: 

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

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

`\excursion`

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

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

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

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

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

`\excursiongroup`

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

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



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

7.4 Representing Problems and Solutions

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

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

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

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

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

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

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

Example 40

Input:

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

`\ifsolutions`

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

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

`\mcc`

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

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

If we start the solutions, then we get

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

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

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

¹²EdNOTE: MK: that did not work!

Example 42

Input:

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

Output:

Problem 7.4.3 (Functions)

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

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

\includeproblem

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

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

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

7.5 Homeworks, Quizzes and Exams

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

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-04-27

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

<u><code>\sref</code></u>	<code>\sref[<i><opt-args></i>]{<i><id></i>}</code>
---------------------------	--

References the label with if *<id>*. Optional arguments: **TODO**

<u><code>\srefsym</code></u>	<code>\srefsym[<i><opt-args></i>]{<i><symbol></i>}</code>
------------------------------	---

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>`.

<u><code>\srefsymuri</code></u>	<code>\srefsymuri{<i><URI></i>}{<i><text></i>}</code>
---------------------------------	---

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\stex_activate_module:n`

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

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

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

12.1.2 Imports and Inheritance

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

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

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

`\l_stex_import_name_str`
`\l_stex_import_archive_str`
`\l_stex_import_path_str`
`\l_stex_import_ns_str`

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

- **name**: An (OMDOC) name. By default equal to $\langle\mathrm{macroname}\rangle$.
- **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. $\backslash\mathrm{symdecl}\{\mathrm{plus}\}[\mathrm{args}=\mathrm{ii}]$ allows for $\backslash\mathrm{plus}\{2\}\{2\}$.
 - a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. $\backslash\mathrm{symdecl}\{\mathrm{plus}\}[\mathrm{args}=\mathrm{a}]$ allows for $\backslash\mathrm{plus}\{2,2,2\}$.
 - 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. $\backslash\mathrm{symdecl}\{\mathrm{forall}\}[\mathrm{args}=\mathrm{bi}]$ allows for $\backslash\mathrm{forall}\{x\in\mathrm{Nat}\}\{x\geq 0\}$.

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Chapter 15

STEX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTEX-Proofs: Structural Markup for Proofs

Chapter 18

sT_EX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

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

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV

Implementation

Chapter 24

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

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

```

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

```

24.2 Preliminaries

```

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

```

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

```

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

```

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

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

116 `<@=stex_annotate>`

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

117 `\tl_new:N \l_stex_html_arg_tl`

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

`_stex_html_checkempty:n`

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

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

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

`\stex_if_do_html:TF`

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

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

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

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

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

`\stex_annotate:bnx`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

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

```

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

24.5 Babel Languages

```

153 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

```

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

```

24.6 Persistence

```

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

```

```

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

```

24.7 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

`\ignorespacesandpars`

```

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

```



```

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

```

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

`\MMTrule`

```

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

```

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

Chapter 25

STEX -MathHub Implementation

```
378 <*package>
379
380 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
381
382 <@@=stex_path>
383
384 Warnings and error messages
385 \msg_new:nnn{stex}{error/norepository}{
386   No~archive~#1~found~in~#2
387 }
388 \msg_new:nnn{stex}{error/notinarchive}{
389   Not~currently~in~an~archive,~but~\detokenize{#1}~
390   needs~one!
391 }
392 \msg_new:nnn{stex}{error/nofile}{
393   \detokenize{#1}~could~not~find~file~#2
394 }
395 \msg_new:nnn{stex}{error/twofiles}{
396   \detokenize{#1}~found~two~candidates~for~#2
397 }
```

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`

```
396 \cs_new_protected:Nn \stex_path_from_string:Nn {
397   \str_set:Nx \l_tmpa_str { #2 }
398   \str_if_empty:NTF \l_tmpa_str {
399     \seq_clear:N #1
400   }{
401     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
402     \sys_if_platform_windows:T{
403       \seq_clear:N \l_tmpa_tl
```

```

404     \seq_map_inline:Nn #1 {
405       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
406       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
407     }
408     \seq_set_eq:NN #1 \l_tmpa_tl
409   }
410   \stex_path_canonicalize:N #1
411 }
412 }
413

```

(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`

```

414 \cs_new_protected:Nn \stex_path_to_string:NN {
415   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
416 }
417
418 \cs_new:Nn \stex_path_to_string:N {
419   \seq_use:Nn #1 /
420 }

```

(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`

```

421 \str_const:Nn \c__stex_path_dot_str {.}
422 \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.

```

423 \cs_new_protected:Nn \stex_path_canonicalize:N {
424   \seq_if_empty:NF #1 {
425     \seq_clear:N \l_tmpa_seq
426     \seq_get_left:NN #1 \l_tmpa_tl
427     \str_if_empty:NT \l_tmpa_tl {
428       \seq_put_right:Nn \l_tmpa_seq {}
429     }
430     \seq_map_inline:Nn #1 {
431       \str_set:Nn \l_tmpa_tl { ##1 }
432       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
433         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
434           \seq_if_empty:NNTF \l_tmpa_seq {
435             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
436               \c__stex_path_up_str
437             }
438           }{
439             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
440             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
441               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
442                 \c__stex_path_up_str
443               }
444             }{

```

```

445         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
446     }
447 }
448 }{
449     \str_if_empty:NF \l_tmpa_tl {
450         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
451     }
452 }
453 }
454 }
455 \seq_gset_eq:NN #1 \l_tmpa_seq
456 }
457 }

```

(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}`

```

458 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
459     \seq_if_empty:NTF #1 {
460         \prg_return_false:
461     }{
462         \seq_get_left:NN #1 \l_tmpa_tl
463         \sys_if_platform_windows:TF{
464             \str_if_in:NnTF \l_tmpa_tl {:}{
465                 \prg_return_true:
466             }{
467                 \prg_return_false:
468             }
469         }{
470             \str_if_empty:NTF \l_tmpa_tl {
471                 \prg_return_true:
472             }{
473                 \prg_return_false:
474             }
475         }
476     }
477 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

478 \str_new:N\l_stex_kpsewhich_return_str
479 \cs_new_protected:Nn \stex_kpsewhich:n {
480     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
481     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
482     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
483 }

```

(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`

```

484 \sys_if_platform_windows:TF{
485   \begingroup\escapechar=-1\catcode'\=12
486   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
487   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
488   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
489   }}{
490   \stex_kpsewhich:n{-var-value~PWD}
491   }
492
493 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
494 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
495 \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

```

496 <@@=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

```

497 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

498 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
499 \stex_path_from_string:Nn \c_stex_mainfile_seq
500 \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`

```

501 \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`

```

502 \cs_new_protected:Nn \stex_filestack_push:n {
503   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
504   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
505     \stex_path_from_string:Nn\g_stex_currentfile_seq{
506       \c_stex_pwd_str/#1
507     }
508   }
509   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
510   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
511 }

```

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

`\stex_filestack_pop:`

```

512 \cs_new_protected:Nn \stex_filestack_pop: {
513   \seq_if_empty:NF\g__stex_files_stack{
514     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
515   }
516   \seq_if_empty:NTF\g__stex_files_stack{
517     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
518   }{
519     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
520     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
521   }
522 }
```

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

Hooks for the current file:

```

523 \AddToHook{file/before}{
524   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
525 }
526 \AddToHook{file/after}{
527   \stex_filestack_pop:
528 }
```

25.4 MathHub Repositories

529 `<@=stex_mathhub>`

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

```

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

```

553     \sys_if_platform_windows:T{
554       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
555     }
556     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
557     \endgroup
558     \ior_close:N \l_tmpa_ior
559   }
560 }
561 \str_if_empty:NTF\c_stex_mathhub_str{
562   \msg_warning:nn{stex}{warning/nomathhub}
563 }{
564   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
565   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
566 }
567 }{
568   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
569   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
570     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
571       \c_stex_pwd_str/\mathhub
572     }
573   }
574   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
575   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
576 }

```

(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

```

577 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
578   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
579     \str_set:Nx \l_tmpa_str { #1 }
580     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
581     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
582     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
583     \_stex_mathhub_find_manifest:N \l_tmpa_seq
584     \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
585       \msg_error:nnxx{stex}{error/norepository}{#1}{
586         \stex_path_to_string:N \c_stex_mathhub_str
587       }
588     } {
589       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
590     }
591   }
592 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```

593 \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`:

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

629 \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:

```

630 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
631   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
632   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
633   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
634     \str_set:Nn \l_tmpa_str {##1}
635     \exp_args:NNoo \seq_set_split:Nnn
636       \l_tmpb_seq \c_colon_str \l_tmpa_str
637     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

638 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
639 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
640 }
641 \exp_args:No \str_case:nnTF \l_tmpa_tl {
642 {id} {
643 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
644 { id } \l_tmpb_tl
645 }
646 {narration-base} {
647 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
648 { narr } \l_tmpb_tl
649 }
650 {url-base} {
651 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
652 { docurl } \l_tmpb_tl
653 }
654 {source-base} {
655 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
656 { ns } \l_tmpb_tl
657 }
658 {ns} {
659 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
660 { ns } \l_tmpb_tl
661 }
662 {dependencies} {
663 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
664 { deps } \l_tmpb_tl
665 }
666 }{}{}
667 }{}
668 }
669 \ior_close:N \c__stex_mathhub_manifest_ior
670 \stex_persist:x {
671 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
672 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
673 }
674 }
675 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

676 \cs_new_protected:Nn \stex_set_current_repository:n {
677 \stex_require_repository:n { #1 }
678 \prop_set_eq:Nc \l_stex_current_repository_prop {
679 c_stex_mathhub_#1_manifest_prop
680 }
681 }

```

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

`\stex_require_repository:n`

```

682 \cs_new_protected:Nn \stex_require_repository:n {
683 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
684 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

685   \_stex_mathhub_do_manifest:n { #1 }
686   }
687 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

688 %\prop_new:N \l_stex_current_repository_prop
689 \bool_if:NF \c_stex_persist_mode_bool {
690   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
691   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
692     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
693   } {
694     \_stex_mathhub_parse_manifest:n { main }
695     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
696     \l_tmpa_str
697     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
698     \c_stex_mathhub_main_manifest_prop
699     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
700     \stex_debug:nn{mathhub}{Current~repository:~
701     \prop_item:Nn \l_stex_current_repository_prop {id}
702   }
703 }
704 }

```

(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.

```

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

```

```

730         no~repository
731     }
732 }
733 \prop_if_exist:NTF \l_stex_current_repository_prop {
734     \stex_set_current_repository:n {
735         \prop_item:Nn \l_stex_current_repository_prop { id }
736     }
737 }{
738     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
739 }
740 }
741 }
742 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

743 \def \mhpath #1 #2 {
744     \exp_args:Ne \tl_if_empty:nTF{#1}{
745         \c_stex_mathhub_str /
746         \prop_item:Nn \l_stex_current_repository_prop { id }
747         / source / #2
748     }{
749         \c_stex_mathhub_str / #1 / source / #2
750     }
751 }

```

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

`\inputref`

`\mhinput`

```

752 \newif \ifinputref \inputreffalse
753
754 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
755     \stex_in_repository:nn {#1} {
756         \ifinputref
757             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
758         \else
759             \inputreftrue
760             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
761         \inputreffalse
762     \fi
763 }
764 }
765 \NewDocumentCommand \mhinput { 0{} m }{
766     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
767 }
768
769 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
770     \stex_in_repository:nn {#1} {
771         \stex_html_backend:TF {
772             \str_clear:N \l_tmpa_str

```

```

773     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
774       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
775     }
776     \stex_annotate_invisible:nnn{inputref}{
777       \l_tmpa_str / #2
778     }{}
779   }{
780     \begingroup
781     \inputreftrue
782     \tl_if_empty:nTF{ ##1 }{
783       \input{#2}
784     }{
785       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
786     }
787     \endgroup
788   }
789 }
790 }
791 \NewDocumentCommand \inputref { 0{} m}{
792   \__stex_mathhub_inputref:nn{ #1 }{} #2 }
793 }

```

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

`\addmhbibresource`

```

794 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
795   \stex_in_repository:nn {#1} {
796     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
797   }
798 }
799 \newcommand\addmhbibresource[2][]{
800   \__stex_mathhub_mhbibresource:nn{ #1 }{} #2 }
801 }

```

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

`\libinput`

```

802 \cs_new_protected:Npn \libinput #1 {
803   \prop_if_exist:NF \l_stex_current_repository_prop {
804     \msg_error:nnn{stex}{error/notinarchive}\libinput
805   }
806   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
807     \msg_error:nnn{stex}{error/notinarchive}\libinput
808   }
809   \seq_clear:N \l__stex_mathhub_libinput_files_seq
810   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
811   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
812
813   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
814     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
815     \IfFileExists{ \l_tmpa_str }{
816       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
817     }{}
818     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
819     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

820 }
821
822 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
823 \IfFileExists{ \l_tmpa_str }{
824   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
825 }{}
826
827 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
828   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
829 }{
830   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
831     \input{ ##1 }
832   }
833 }
834 }

```

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

`\libusepackage`

```

835 \NewDocumentCommand \libusepackage {0{} m} {
836   \prop_if_exist:NF \l_stex_current_repository_prop {
837     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
838   }
839   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
840     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
841   }
842   \seq_clear:N \l__stex_mathhub_libinput_files_seq
843   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
844   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
845
846   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
847     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
848     \IfFileExists{ \l_tmpa_str.sty }{
849       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
850     }{}
851     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
852     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
853   }
854
855   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
856   \IfFileExists{ \l_tmpa_str.sty }{
857     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
858   }{}
859
860   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
861     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
862   }{
863     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
864       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
865         \usepackage[#1]{ ##1 }
866       }
867     }{
868       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
869     }
870   }

```

```

870 }
871 }

```

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

```

\mhgraphics
\cmhgraphics

```

```

872
873 \AddToHook{begindocument}{
874 \ltx@ifpackageloaded{graphicx}{
875   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
876   \newcommand\mhgraphics[2][]{%
877     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
878     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
879   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
880 }{}

```

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

```

\lstinputmhlisting
\clstinputmhlisting

```

```

881 \ltx@ifpackageloaded{listings}{
882   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
883   \newcommand\lstinputmhlisting[2][]{%
884     \def\lst@mhrepos{}\setkeys{lst}{#1}%
885     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
886   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
887 }{}
888 }
889
890 \</package>

```

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

Chapter 26

STEX -References Implementation

```
891 <*package>
892
893 %%%%%%%%%% references.dtx %%%%%%%%%%
894
895 <@@=stex_refs>
      Warnings and error messages
896
```

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

```
897 %\iow_new:N \c__stex_refs_refs_iow
898 \AtBeginDocument{
899 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
900 }
901 \AtEndDocument{
902 % \iow_close:N \c__stex_refs_refs_iow
903 }
```

`\STEXreftitle`

```
904 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
905
906 \NewDocumentCommand \STEXreftitle { m } {
907   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
908 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
909 \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:`

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

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

`\l_stex_current_docurl_str`

```
950 \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:`

```
951 \cs_new_protected:Nn \stex_get_document_url: {
952   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
953   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
954   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

955 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
956 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
957
958 \str_clear:N \l_tmpa_str
959 \prop_if_exist:NT \l_stex_current_repository_prop {
960   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
961     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
962       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
963     }
964   }
965 }
966
967 \str_if_empty:NTF \l_tmpa_str {
968   \str_set:Nx \l_stex_current_docurl_str {
969     file:/\stex_path_to_string:N \l_tmpa_seq
970   }
971 }{
972   \bool_set_true:N \l_tmpa_bool
973   \bool_while_do:Nn \l_tmpa_bool {
974     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
975     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
976       {source} { \bool_set_false:N \l_tmpa_bool }
977     }{}{
978       \seq_if_empty:NT \l_tmpa_seq {
979         \bool_set_false:N \l_tmpa_bool
980       }
981     }
982   }
983
984   \seq_if_empty:NTF \l_tmpa_seq {
985     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
986   }{
987     \str_set:Nx \l_stex_current_docurl_str {
988       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
989     }
990   }
991 }
992 }

```

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

26.2 Setting Reference Targets

```

993 \str_const:Nn \c__stex_refs_url_str{URL}
994 \str_const:Nn \c__stex_refs_ref_str{REF}
995 \str_new:N \l__stex_refs_curr_label_str
996 % @currentlabel -> number
997 % @currentlabelname -> title
998 % @currentHref -> name.number <- id of some kind
999 % \theH# -> \arabic{section}
1000 % \the# -> number
1001 % \hyper@makecurrent{#}
1002 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1003 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1004   \stex_get_document_uri:
1005   \str_clear:N \l__stex_refs_curr_label_str
1006   \str_set:Nx \l_tmpa_str { #1 }
1007   \str_if_empty:NT \l_tmpa_str {
1008     \int_incr:N \l__stex_refs_unnamed_counter_int
1009     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1010   }
1011   \str_set:Nx \l__stex_refs_curr_label_str {
1012     \l_stex_current_docns_str?\l_tmpa_str
1013   }
1014   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1015     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1016   }
1017   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1018     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1019   }
1020   \stex_if_smsmode:TF {
1021     \stex_get_document_url:
1022     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1023     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1024   }{
1025     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1026     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1027     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1028     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1029   }
1030 }

```

(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.

```

1031 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1032   \str_set:Nn \l_tmpa_str {#1?#2}
1033   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1034   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1035     \seq_new:c {g__stex_refs_labels_#2_seq}
1036   }
1037   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1038     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1039   }
1040 }

```

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

```

1041 \AtEndDocument{
1042   \def\stexauxadddocref#1 #2 {}{}
1043 }

```

`\stex_ref_new_sym_target:n`

```

1044 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1045   \stex_if_smsmode:TF {
1046     \str_if_exist:cF{sref_sym_#1_type}{
1047       \stex_get_document_url:
1048       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1049     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1050   }
1051   }{
1052     \str_if_empty:NF \l__stex_refs_curr_label_str {
1053       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1054       \immediate\write\@auxout{
1055         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1056           \l__stex_refs_curr_label_str
1057         }
1058       }
1059     }
1060   }
1061 }

```

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

26.3 Using References

```

1062 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1063
1064 \keys_define:nn { stex / sref } {
1065   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1066   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1067   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1068   post          .tl_set:N = \l__stex_refs_post_tl ,
1069 }
1070 \cs_new_protected:Nn \__stex_refs_args:n {
1071   \tl_clear:N \l__stex_refs_linktext_tl
1072   \tl_clear:N \l__stex_refs_fallback_tl
1073   \tl_clear:N \l__stex_refs_pre_tl
1074   \tl_clear:N \l__stex_refs_post_tl
1075   \str_clear:N \l__stex_refs_repo_str
1076   \keys_set:nn { stex / sref } { #1 }
1077 }

```

The actual macro:

```

1078 \NewDocumentCommand \sref { 0{} m}{
1079   \__stex_refs_args:n { #1 }
1080   \str_if_empty:NTF \l__stex_refs_indocument_str {
1081     \str_set:Nx \l_tmpa_str { #2 }
1082     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1083     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1084       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1085         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1086           \str_clear:N \l_tmpa_str
1087         }
1088       }{
1089         \str_clear:N \l_tmpa_str
1090       }
1091     }{
1092       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1093       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1094 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1095 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1096   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1097   \str_clear:N \l_tmpa_str
1098   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1099     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1100       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1101     }{
1102       \seq_map_break:n {
1103         \str_set:Nn \l_tmpa_str { ##1 }
1104       }
1105     }
1106   }
1107 }{
1108   \str_clear:N \l_tmpa_str
1109 }
1110 }
1111 \str_if_empty:NTF \l_tmpa_str {
1112   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1113 }{
1114   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1115     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1116       \cs_if_exist:cTF{autoref}{
1117         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1118       }{
1119         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1120       }
1121     }{
1122       \ltx@ifpackageloaded{hyperref}{
1123         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1124       }{
1125         \l__stex_refs_linktext_tl
1126       }
1127     }
1128   }{
1129     \ltx@ifpackageloaded{hyperref}{
1130       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1131     }{
1132       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1133     }
1134   }
1135 }
1136 }{
1137   % TODO
1138 }
1139 }

```

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

`\srefsym`

```

1140 \NewDocumentCommand \srefsym { 0{} m}{
1141   \stex_get_symbol:n { #2 }
1142   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1143 }

```

```

1144
1145 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1146   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1147     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1148   }{
1149     \__stex_refs_args:n { #1 }
1150     \str_if_empty:NTF \l__stex_refs_indocument_str {
1151       \tl_if_exist:cTF{sref_sym_#2 _type}{
1152         % doc uri in \l_tmpb_str
1153         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1154         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1155           % reference
1156           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1157             \cs_if_exist:cTF{autoref}{
1158               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1159             }{
1160               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1161             }
1162           }{
1163             \ltx@ifpackageloaded{hyperref}{
1164               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1165             }{
1166               \l__stex_refs_linktext_tl
1167             }
1168           }
1169         }{
1170           % URL
1171           \ltx@ifpackageloaded{hyperref}{
1172             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1173           }{
1174             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1175           }
1176         }
1177       }{
1178         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1179       }
1180     }{
1181       % TODO
1182     }
1183   }
1184 }

```

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

\srefsymuri

```

1185 \cs_new_protected:Npn \srefsymuri #1 #2 {
1186   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1187 }

```

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

```

1188 </package>

```

Chapter 27

STEX -Modules Implementation

```
1189 <*package>
1190
1191 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1192
1193 <@@=stex_modules>
1194
1195     Warnings and error messages
1196 \msg_new:nnn{stex}{error/unknownmodule}{
1197     No~module~#1~found
1198 }
1199 \msg_new:nnn{stex}{error/syntax}{
1200     Syntax~error:~#1
1201 }
1202 \msg_new:nnn{stex}{error/siglanguage}{
1203     Module~#1~declares~signature~#2,~but~does~not~
1204     declare~its~language
1205 }
1206 \msg_new:nnn{stex}{warning/deprecated}{
1207     #1~is~deprecated;~please~use~#2~instead!
1208 }
1209 \msg_new:nnn{stex}{error/conflictingmodules}{
1210     Conflicting~imports~for~module~#1
1211 }
1212
1213 \l_stex_current_module_str The current module:
1214 \str_new:N \l_stex_current_module_str
1215
1216 (End definition for \l_stex_current_module_str. This variable is documented on page 71.)
1217
1218 \l_stex_all_modules_seq Stores all available modules
1219 \seq_new:N \l_stex_all_modules_seq
1220
1221 (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
1213 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1214   \str_if_empty:NTF \l_stex_current_module_str
1215   \prg_return_false: \prg_return_true:
1216 }

(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
1217 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1218   \prop_if_exist:cTF { c_stex_module_#1_prop }
1219   \prg_return_true: \prg_return_false:
1220 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1221 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1222   \stex_add_to_current_module:n { #1 }
1223   \stex_do_up_to_module:n { #1 }
1224 }}
1225 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1226
1227 \cs_new_protected:Nn \stex_add_to_current_module:n {
1228   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1229 }
1230 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1231 \cs_new_protected:Npn \STEXexport {
1232   \begingroup
1233   \newlinechar=-1\relax
1234   \endlinechar=-1\relax
1235   %\catcode'\ = 9\relax
1236   \expandafter\endgroup\__stex_modules_export:n
1237 }
1238 \cs_new_protected:Nn \__stex_modules_export:n {
1239   \ignorespaces #1
1240   \stex_add_to_current_module:n { \ignorespaces #1 }
1241   \stex_smsmode_do:
1242 }
1243 \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
1244 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1245   \str_set:Nx \l_tmpa_str { #1 }
1246   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1247 }

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

```


`\stex_add_import_to_current_module:n`

```

1248 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1249   \str_set:Nx \l_tmpa_str { #1 }
1250   \exp_args:Nno
1251   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1252     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1253   }
1254 }

```

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

`\stex_collect_imports:n`

```

1255 \cs_new_protected:Nn \stex_collect_imports:n {
1256   \seq_clear:N \l_stex_collect_imports_seq
1257   \__stex_modules_collect_imports:n {#1}
1258 }
1259 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1260   \seq_map_inline:cn {c_stex_module_#1_imports} {
1261     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1262       \__stex_modules_collect_imports:n { ##1 }
1263     }
1264   }
1265   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1266     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1267   }
1268 }

```

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

`\stex_do_up_to_module:n`

```

1269 \int_new:N \l__stex_modules_group_depth_int
1270 \cs_new_protected:Nn \stex_do_up_to_module:n {
1271   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1272     #1
1273   }{
1274     #1
1275     \expandafter \tl_gset:Nn
1276     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1277     \expandafter\expandafter\expandafter\endcsname
1278     \expandafter\expandafter\expandafter { \csname
1279       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1280     \aftergroup\__stex_modules_aftergroup_do:
1281   }
1282 }
1283 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1284 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1285   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1286     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1287   }}
1288   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1289     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1290     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1291   }{
1292     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```

```

1293 \aftergroup\__stex_modules_aftergroup_do:
1294 }
1295 }
1296 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1297 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1298 }

```

(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).

```

1299

```

(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.

```

1300 \str_new:N \l_stex_module_ns_str
1301 \str_new:N \l_stex_module_subpath_str
1302 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1303 \seq_set_eq:NN \l_tmpa_seq #2
1304 % split off file extension
1305 \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1306 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1307 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1308 \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1309
1310 \bool_set_true:N \l_tmpa_bool
1311 \bool_while_do:Nn \l_tmpa_bool {
1312 \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1313 \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1314 {source} { \bool_set_false:N \l_tmpa_bool }
1315 }{}{
1316 \seq_if_empty:NT \l_tmpa_seq {
1317 \bool_set_false:N \l_tmpa_bool
1318 }
1319 }
1320 }
1321
1322 \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1323 % \l_tmpa_seq <- sub-path relative to archive
1324 \str_if_empty:NTF \l_stex_module_subpath_str {
1325 \str_set:Nx \l_stex_module_ns_str {#1}
1326 }{
1327 \str_set:Nx \l_stex_module_ns_str {
1328 #1/\l_stex_module_subpath_str
1329 }
1330 }
1331 }
1332
1333 \cs_new_protected:Nn \stex_modules_current_namespace: {
1334 \str_clear:N \l_stex_module_subpath_str
1335 \prop_if_exist:NTF \l_stex_current_repository_prop {
1336 \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1337     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1338   }{
1339     % split off file extension
1340     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1341     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1342     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1343     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1344     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1345     \str_set:Nx \l_stex_module_ns_str {
1346       file:/\stex_path_to_string:N \l_tmpa_seq
1347     }
1348   }
1349 }

```

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

27.1 The smodule environment

smodule arguments:

```

1350 \keys_define:nn { stex / module } {
1351   title      .tl_set:N      = \smodulename ,
1352   type       .str_set_x:N   = \smodulename ,
1353   id         .str_set_x:N   = \smoduleid ,
1354   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1355   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1356   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1357   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1358   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1359   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1360   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1361   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1362 }
1363
1364 \cs_new_protected:Nn \__stex_modules_args:n {
1365   \str_clear:N \smodulename
1366   \str_clear:N \smoduleid
1367   \str_clear:N \l_stex_module_ns_str
1368   \str_clear:N \l_stex_module_deprecate_str
1369   \str_clear:N \l_stex_module_lang_str
1370   \str_clear:N \l_stex_module_sig_str
1371   \str_clear:N \l_stex_module_creators_str
1372   \str_clear:N \l_stex_module_contributors_str
1373   \str_clear:N \l_stex_module_meta_str
1374   \str_clear:N \l_stex_module_srccite_str
1375   \keys_set:nn { stex / module } { #1 }
1376 }
1377
1378 % module parameters here? In the body?
1379
1380

```

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

```

1381 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1382 \int_set:Nn \l_stex_modules_group_depth_int {\currentgrouplevel}
1383 \str_set:Nx \l_stex_module_name_str { #2 }
1384 \__stex_modules_args:n { #1 }

```

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

```

1385 \stex_if_in_module:TF {
1386   % Nested module
1387   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1388   { ns } \l_stex_module_ns_str
1389   \str_set:Nx \l_stex_module_name_str {
1390     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1391     { name } / \l_stex_module_name_str
1392   }
1393   \str_if_empty:NT \l_stex_module_lang_str {
1394     \str_set:Nx \l_stex_module_lang_str {
1395       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1396       { lang }
1397     }
1398   }
1399 }{
1400   % not nested:
1401   \str_if_empty:NT \l_stex_module_ns_str {
1402     \stex_modules_current_namespace:
1403     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1404       / {\l_stex_module_ns_str}
1405     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1406     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1407       \str_set:Nx \l_stex_module_ns_str {
1408         \stex_path_to_string:N \l_tmpa_seq
1409       }
1410     }
1411   }
1412 }

```

Next, we determine the language of the module:

```

1413 \str_if_empty:NT \l_stex_module_lang_str {
1414   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1415   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1416   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1417   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1418     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1419       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1420     }
1421   }
1422   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1423   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1424     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1425     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1426       inferred~from~file~name}
1427   }
1428 }
1429
1430 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1431 \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1432 }}

```

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

```

1433 \str_if_empty:NTF \l_stex_module_sig_str {
1434   \exp_args:Nnx \prop_gset_from_keyval:cn {
1435     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1436   } {
1437     name      = \l_stex_module_name_str ,
1438     ns        = \l_stex_module_ns_str ,
1439     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1440     lang      = \l_stex_module_lang_str ,
1441     sig       = \l_stex_module_sig_str ,
1442     deprecate = \l_stex_module_deprecate_str ,
1443     meta      = \l_stex_module_meta_str
1444   }
1445   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1446   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1447   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1448   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1449   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1450 \str_if_empty:NT \l_stex_module_meta_str {
1451   \str_set:Nx \l_stex_module_meta_str {
1452     \c_stex_metatheory_ns_str ? Metatheory
1453   }
1454 }
1455 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1456   \bool_set_true:N \l_stex_in_meta_bool
1457   \exp_args:Nx \stex_add_to_current_module:n {
1458     \bool_set_true:N \l_stex_in_meta_bool
1459     \stex_activate_module:n {\l_stex_module_meta_str}
1460     \bool_set_false:N \l_stex_in_meta_bool
1461   }
1462   \stex_activate_module:n {\l_stex_module_meta_str}
1463   \bool_set_false:N \l_stex_in_meta_bool
1464 }
1465 }{
1466   \str_if_empty:NT \l_stex_module_lang_str {
1467     \msg_error:nnxx{stex}{error/siglanguage}{
1468       \l_stex_module_ns_str?\l_stex_module_name_str
1469     }\l_stex_module_sig_str}
1470   }
1471   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1472   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1473     \stex_debug:nn{modules}{(already exists)}
1474   }{
1475     \stex_debug:nn{modules}{(needs loading)}
1476     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1477     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1478     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1479     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex

```

```

1480 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1481 \str_set:Nx \l_tmpa_str {
1482   \stex_path_to_string:N \l_tmpa_seq /
1483   \l_tmpa_str . \l_stex_module_sig_str .tex
1484 }
1485 \IfFileExists \l_tmpa_str {
1486   \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1487     \str_clear:N \l_stex_current_module_str
1488     \seq_clear:N \l_stex_all_modules_seq
1489     \stex_debug:nn{modules}{Loading~signature}
1490   }
1491 }{
1492   \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1493 }
1494 }
1495 \stex_if_smsmode:F {
1496   \stex_activate_module:n {
1497     \l_stex_module_ns_str ? \l_stex_module_name_str
1498   }
1499 }
1500 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1501 }
1502 \str_if_empty:NF \l_stex_module_deprecate_str {
1503   \msg_warning:nnxx{stex}{warning/deprecated}{
1504     Module~\l_stex_current_module_str
1505   }{
1506     \l_stex_module_deprecate_str
1507   }
1508 }
1509 \seq_put_right:Nx \l_stex_all_modules_seq {
1510   \l_stex_module_ns_str ? \l_stex_module_name_str
1511 }
1512 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1513 }

```

(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}

1514 \cs_new_protected:Nn \__stex_modules_begin_module: {
1515   \stex_reactivate_macro:N \STEXexport
1516   \stex_reactivate_macro:N \importmodule
1517   \stex_reactivate_macro:N \symdecl
1518   \stex_reactivate_macro:N \notation
1519   \stex_reactivate_macro:N \symdef
1520
1521   \stex_debug:nn{modules}{
1522     New~module:\\
1523     Namespace:~\l_stex_module_ns_str\\
1524     Name:~\l_stex_module_name_str\\
1525     Language:~\l_stex_module_lang_str\\
1526     Signature:~\l_stex_module_sig_str\\
1527     Metatheory:~\l_stex_module_meta_str\\

```

```

1528   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1529 }
1530
1531 \stex_if_do_html:T{
1532   \begin{stex_annotate_env} {theory} {
1533     \l_stex_module_ns_str ? \l_stex_module_name_str
1534   }
1535
1536   \stex_annotate_invisible:nnn{header}{} {
1537     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1538     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1539     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1540       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1541     }
1542     \str_if_empty:NF \smoduletype {
1543       \stex_annotate:nnn{type}{\smoduletype}{}
1544     }
1545   }
1546 }
1547 % TODO: Inherit metatheory for nested modules?
1548 }
1549 \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}

```

1550 \cs_new_protected:Nn \_stex_modules_end_module: {
1551   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_str}
1552   \stex_reset_up_to_module:n \l_stex_current_module_str
1553   \stex_if_smsmode:T {
1554     \stex_persist:x {
1555       \prop_set_from_keyval:cn{c_stex_module_\l_stex_current_module_str _prop}{
1556         \exp_after:wN \prop_to_keyval:N \csname c_stex_module_\l_stex_current_module_str _pr
1557       }
1558       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _constants}{
1559         \seq_use:cn{c_stex_module_\l_stex_current_module_str _constants},
1560       }
1561       \seq_set_from_clist:cn{c_stex_module_\l_stex_current_module_str _imports}{
1562         \seq_use:cn{c_stex_module_\l_stex_current_module_str _imports},
1563       }
1564       \tl_set:cn {c_stex_module_\l_stex_current_module_str _code}
1565     }
1566     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module_\l_stex_current_module_str
1567     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1568   }
1569 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1570 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1571 \NewDocumentEnvironment { smodule } { 0 } { m } {
1572   \stex_module_setup:nn{#1}{#2}
1573   \par
1574   \stex_if_smsmode:F{

```

```

1575 \tl_if_empty:NF \smodulename {
1576 \exp_args:No \stex_document_title:n \smodulename
1577 }
1578 \tl_clear:N \l_tmpa_tl
1579 \clist_map_inline:Nn \smodulename {
1580 \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1581 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1582 }
1583 }
1584 \tl_if_empty:NTF \l_tmpa_tl {
1585 \__stex_modules_smodule_start:
1586 }{
1587 \l_tmpa_tl
1588 }
1589 }
1590 \__stex_modules_begin_module:
1591 \str_if_empty:NF \smoduleid {
1592 \stex_ref_new_doc_target:n \smoduleid
1593 }
1594 \stex_smsmode_do:
1595 } {
1596 \__stex_modules_end_module:
1597 \stex_if_smsmode:F {
1598 \end{stex_annotate_env}
1599 \clist_set:No \l_tmpa_clist \smodulename
1600 \tl_clear:N \l_tmpa_tl
1601 \clist_map_inline:Nn \l_tmpa_clist {
1602 \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1603 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1604 }
1605 }
1606 \tl_if_empty:NTF \l_tmpa_tl {
1607 \__stex_modules_smodule_end:
1608 }{
1609 \l_tmpa_tl
1610 }
1611 }
1612 }

```

\stexpatchmodule

```

1613 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1614 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1615
1616 \newcommand\stexpatchmodule[3] [] {
1617 \str_set:Nx \l_tmpa_str{ #1 }
1618 \str_if_empty:NTF \l_tmpa_str {
1619 \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1620 \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1621 }{
1622 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1623 \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1624 }
1625 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1626 \NewDocumentCommand \STEXModule { m } {
1627   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1628   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1629   \tl_set:Nn \l_tmpa_tl {
1630     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1631   }
1632   \seq_map_inline:Nn \l_stex_all_modules_seq {
1633     \str_set:Nn \l_tmpb_str { ##1 }
1634     \str_if_eq:eeT { \l_tmpa_str } {
1635       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1636     } {
1637       \seq_map_break:n {
1638         \tl_set:Nn \l_tmpa_tl {
1639           \stex_invoke_module:n { ##1 }
1640         }
1641       }
1642     }
1643   }
1644   \l_tmpa_tl
1645 }
1646
1647 \cs_new_protected:Nn \stex_invoke_module:n {
1648   \stex_debug:nn{modules}{Invoking~module~#1}
1649   \peek_charcode_remove:NTF ! {
1650     \__stex_modules_invoke_uri:nN { #1 }
1651   } {
1652     \peek_charcode_remove:NTF ? {
1653       \__stex_modules_invoke_symbol:nn { #1 }
1654     } {
1655       \msg_error:nnx{stex}{error/syntax}{
1656         ?~or~!~expected~after~
1657         \c_backslash_str STEXModule{#1}
1658       }
1659     }
1660   }
1661 }
1662
1663 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1664   \str_set:Nn #2 { #1 }
1665 }
1666
1667 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1668   \stex_invoke_symbol:n{#1?#2}
1669 }

```

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

```

\stex_activate_module:n
1670 \bool_new:N \l_stex_in_meta_bool
1671 \bool_set_false:N \l_stex_in_meta_bool

```

```

1672 \cs_new_protected:Nn \stex_activate_module:n {
1673   \stex_debug:nn{modules}{Activating~module~#1}
1674   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1675     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1676     \use:c{ c_stex_module_#1_code }
1677   }
1678 }

```

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

```

1679 \endpackage

```

Chapter 28

STEX -Module Inheritance Implementation

```
1680 <*package>
1681
1682 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1683
```

28.1 SMS Mode

```
1684 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1685 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1686 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1687 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1688
1689 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1690   \makeatletter
1691   \makeatother
1692   \ExplSyntaxOn
1693   \ExplSyntaxOff
1694   \rustexBREAK
1695 }
1696
1697 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1698   \symdef
1699   \importmodule
1700   \notation
1701   \symdecl
1702   \STEXexport
1703   \inlineass
1704   \inlinedef
1705   \inlineex
1706   \endinput
1707   \setnotation
```

```

1708 \copynotation
1709 \assign
1710 \renamedekl
1711 \donotcopy
1712 \instantiate
1713 }
1714
1715 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1716   \tl_to_str:n {
1717     smodule,
1718     copymodule,
1719     interpretmodule,
1720     sdefinition,
1721     sexample,
1722     sassertion,
1723     sparagraph,
1724     mathstructure
1725   }
1726 }

```

(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`

```

1727 \bool_new:N \g__stex_smsmode_bool
1728 \bool_set_false:N \g__stex_smsmode_bool
1729 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1730   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1731 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1732 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1733   \vbox_set:Nn \l_tmpa_box {
1734     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1735     \bool_gset_true:N \g__stex_smsmode_bool
1736     #2
1737     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1738   }
1739   \box_clear:N \l_tmpa_box
1740 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1741 \quark_new:N \q__stex_smsmode_break
1742
1743 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1744   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2}}
1745   \stex_smsmode_do:
1746 }
1747
1748 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1749   \_stex_modules_args:n{#1}

```

```

1750 \stex_if_in_module:F {
1751   \str_if_empty:NF \l_stex_module_sig_str {
1752     \stex_modules_current_namespace:
1753     \str_set:Nx \l_stex_module_name_str { #2 }
1754     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1755       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1756       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1757       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1758       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1759       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1760       \str_set:Nx \l_tmpa_str {
1761         \stex_path_to_string:N \l_tmpa_seq /
1762         \l_tmpa_str . \l_stex_module_sig_str .tex
1763       }
1764       \IfFileExists \l_tmpa_str {
1765         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1766       }{
1767         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1768       }
1769     }
1770   }
1771 }
1772 }
1773
1774 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1775   \stex_filestack_push:n{#1}
1776   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1777   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1778   % ----- new -----
1779   \__stex_smsmode_in_smsmode:nn{#1}{
1780     \let\importmodule\__stex_smsmode_importmodule:
1781     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1782     \let\__stex_modules_begin_module:\relax
1783     \let\__stex_modules_end_module:\relax
1784     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1785     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1786     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1787     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1788     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1789     \everyeof{\q__stex_smsmode_break\noexpand}
1790     \expandafter\expandafter\expandafter
1791     \stex_smsmode_do:
1792     \csname @ @ input\endcsname "#1"\relax
1793
1794     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1795       \stex_filestack_push:n{##1}
1796       \expandafter\expandafter\expandafter
1797       \stex_smsmode_do:
1798       \csname @ @ input\endcsname "##1"\relax
1799       \stex_filestack_pop:
1800     }
1801   }
1802   % ----- new -----
1803   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1804 #2
1805 % ----- new -----
1806 \begingroup
1807 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1808 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1809   \stex_import_module_uri:nn ##1
1810   \stex_import_require_module:nnnn
1811   \l_stex_import_ns_str
1812   \l_stex_import_archive_str
1813   \l_stex_import_path_str
1814   \l_stex_import_name_str
1815 }
1816 \endgroup
1817 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1818 % ----- new -----
1819 \everyeof{\q__stex_smsmode_break\noexpand}
1820 \expandafter\expandafter\expandafter
1821 \stex_smsmode_do:
1822 \csname @ @ input\endcsname "#1"\relax
1823 }
1824 \stex_filestack_pop:
1825 }

```

(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:

```

1826 \cs_new_protected:Npn \stex_smsmode_do: {
1827   \stex_if_smsmode:T {
1828     \__stex_smsmode_do:w
1829   }
1830 }
1831 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1832   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1833     \expandafter\if\expandafter\relax\noexpand#1
1834     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1835     \else\expandafter\__stex_smsmode_do:w\fi
1836   }{
1837     \__stex_smsmode_do:w %#1
1838   }
1839 }
1840 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1841   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1842     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1843       #1\__stex_smsmode_do:w
1844     }{
1845       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1846         #1
1847       }{
1848         \cs_if_eq:NNTF \begin #1 {
1849           \__stex_smsmode_check_begin:n
1850         }{
1851           \cs_if_eq:NNTF \end #1 {
1852             \__stex_smsmode_check_end:n

```

```

1853         }{
1854             \__stex_smsmode_do:w
1855         }
1856     }
1857 }
1858 }
1859 }
1860 }
1861
1862 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1863     \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1864         \begin{#1}
1865     }{
1866         \__stex_smsmode_do:w
1867     }
1868 }
1869 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1870     \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1871         \end{#1}\__stex_smsmode_do:w
1872     }{
1873         \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1874     }
1875 }

```

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

28.2 Inheritance

```

1876 <@@=stex_importmodule>

\stex_import_module_uri:nn

1877 \cs_new_protected:Nn \stex_import_module_uri:nn {
1878     \str_set:Nx \l_stex_import_archive_str { #1 }
1879     \str_set:Nn \l_stex_import_path_str { #2 }
1880
1881     \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1882     \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1883     \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1884
1885     \stex_modules_current_namespace:
1886     \bool_lazy_all:nTF {
1887         {\str_if_empty_p:N \l_stex_import_archive_str}
1888         {\str_if_empty_p:N \l_stex_import_path_str}
1889         {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1890     }{
1891         \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1892         \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1893     }{
1894         \str_if_empty:NT \l_stex_import_archive_str {
1895             \prop_if_exist:NT \l_stex_current_repository_prop {
1896                 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1897             }
1898         }
1899         \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1900     \str_if_empty:NF \l_stex_import_path_str {
1901         \str_set:Nx \l_stex_import_ns_str {
1902             \l_stex_module_ns_str / \l_stex_import_path_str
1903         }
1904     }
1905 }{
1906     \stex_require_repository:n \l_stex_import_archive_str
1907     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1908     \l_stex_import_ns_str
1909     \str_if_empty:NF \l_stex_import_path_str {
1910         \str_set:Nx \l_stex_import_ns_str {
1911             \l_stex_import_ns_str / \l_stex_import_path_str
1912         }
1913     }
1914 }
1915 }
1916 }

```

(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 1917 \str_new:N \l_stex_import_name_str
\l_stex_import_path_str    1918 \str_new:N \l_stex_import_archive_str
\l_stex_import_ns_str      1919 \str_new:N \l_stex_import_path_str
                            1920 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1921 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1922     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1923
1924         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1925
1926         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1927         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1928
1929         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1930
1931         % archive
1932         \str_set:Nx \l_tmpa_str { #2 }
1933         \str_if_empty:NTF \l_tmpa_str {
1934             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1935         } {
1936             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1937             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1938             \seq_put_right:Nn \l_tmpa_seq { source }
1939         }
1940
1941         % path
1942         \str_set:Nx \l_tmpb_str { #3 }
1943         \str_if_empty:NTF \l_tmpb_str {
1944             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1945         }

```



```

1946 \ltx@ifpackageloaded{babel} {
1947   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1948     { \language } \l_tmpb_str {
1949       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1950     }
1951   } {
1952     \str_clear:N \l_tmpb_str
1953   }
1954
1955   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1956   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1957     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1958   }{
1959     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1960     \IfFileExists{ \l_tmpa_str.tex }{
1961       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1962     }{
1963       % try english as default
1964       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1965       \IfFileExists{ \l_tmpa_str.en.tex }{
1966         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1967       }{
1968         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1969       }
1970     }
1971   }
1972
1973 } {
1974   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1975   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1976
1977   \ltx@ifpackageloaded{babel} {
1978     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1979       { \language } \l_tmpb_str {
1980         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1981       }
1982   } {
1983     \str_clear:N \l_tmpb_str
1984   }
1985
1986   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1987
1988   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1989   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1990     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1991   }{
1992     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1993     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1994       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1995     }{
1996       % try english as default
1997       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1998       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1999         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```

```

2000     }{
2001     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
2002     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
2003         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
2004     }{
2005         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
2006         \IfFileExists{ \l_tmpa_str.tex }{
2007             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
2008         }{
2009             % try english as default
2010             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
2011             \IfFileExists{ \l_tmpa_str.en.tex }{
2012                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
2013             }{
2014                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
2015             }
2016         }
2017     }
2018 }
2019 }
2020 }
2021 }
2022
2023 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
2024     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
2025         \seq_clear:N \l_stex_all_modules_seq
2026         \str_clear:N \l_stex_current_module_str
2027         \str_set:Nx \l_tmpb_str { #2 }
2028         \str_if_empty:NF \l_tmpb_str {
2029             \stex_set_current_repository:n { #2 }
2030         }
2031         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2032     }
2033
2034     \stex_if_module_exists:nF { #1 ? #4 } {
2035         \msg_error:nnx{stex}{error/unknownmodule}{
2036             #1?#4~(in~file~\g__stex_importmodule_file_str)
2037         }
2038     }
2039 }
2040
2041 }
2042 \stex_activate_module:n { #1 ? #4 }
2043 }

```

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

`\importmodule`

```

2044 \NewDocumentCommand \importmodule { 0{} m } {
2045     \stex_import_module_uri:nn { #1 } { #2 }
2046     \stex_debug:nn{modules}{Importing~module:~
2047         \l_stex_import_ns_str ? \l_stex_import_name_str
2048     }
2049     \stex_import_require_module:nnnn

```

```

2050 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2051 { \l_stex_import_path_str } { \l_stex_import_name_str }
2052 \stex_if_smsmode:F {
2053   \stex_annotate_invisible:nnn
2054   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2055 }
2056 \exp_args:Nx \stex_add_to_current_module:n {
2057   \stex_import_require_module:nnnn
2058   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2059   { \l_stex_import_path_str } { \l_stex_import_name_str }
2060 }
2061 \exp_args:Nx \stex_add_import_to_current_module:n {
2062   \l_stex_import_ns_str ? \l_stex_import_name_str
2063 }
2064 \stex_smsmode_do:
2065 \ignorespacesandpars
2066 }
2067 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

2068 \NewDocumentCommand \usemodule { 0{} m } {
2069   \stex_if_smsmode:F {
2070     \stex_import_module_uri:nn { #1 } { #2 }
2071     \stex_import_require_module:nnnn
2072     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2073     { \l_stex_import_path_str } { \l_stex_import_name_str }
2074     \stex_annotate_invisible:nnn
2075     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2076   }
2077   \stex_smsmode_do:
2078   \ignorespacesandpars
2079 }

```

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

```

2080 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2081   \tl_if_empty:nF{#2}{
2082     \clist_set:Nn \l_tmpa_clist {#2}
2083     \clist_map_inline:Nn \l_tmpa_clist {
2084       \tl_if_head_eq_charcode:nNTF {##1}[{
2085         #1 ##1
2086       }{
2087         #1{##1}
2088       }
2089     }
2090   }
2091 }
2092 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2093
2094
2095 </package>

```

Chapter 29

STEX -Symbols Implementation

```
2096 <*package>
2097
2098 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
2099
    Warnings and error messages
2100 \msg_new:nnn{stex}{error/wrongargs}{
2101   args~value~in~symbol~declaration~for~#1~
2102   needs~to~be~i,~a,~b~or~B,~but~#2~given
2103 }
2104 \msg_new:nnn{stex}{error/unknownsymbol}{
2105   No~symbol~#1~found!
2106 }
2107 \msg_new:nnn{stex}{error/seqlength}{
2108   Expected~#1~arguments;~got~#2!
2109 }
2110 \msg_new:nnn{stex}{error/unknownnotation}{
2111   Unknown~notation~#1~for~#2!
2112 }
```

29.1 Symbol Declarations

```
2113 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2114 \cs_new_protected:Nn \stex_all_symbols:n {
2115   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2116   \seq_map_inline:Nn \l_stex_all_modules_seq {
2117     \seq_map_inline:cn{c_stex_module_##1_constants}{
2118       \__stex_symdecl_all_symbols_cs{##1?####1}
2119     }
2120   }
2121 }
```

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

\STEXsymbol

```
2122 \NewDocumentCommand \STEXsymbol { m } {
2123   \stex_get_symbol:n { #1 }
2124   \exp_args:No
2125   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2126 }
```

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

symdecl arguments:

```
2127 \keys_define:nn { stex / symdecl } {
2128   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2129   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2130   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2131   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2132   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2133   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2134   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2135   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2136   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2137   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2138   assoc     .choices:nn =
2139     {bin,binl,binr,pre,conj,pwconj}
2140     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2141 }
2142
2143 \bool_new:N \l_stex_symdecl_make_macro_bool
2144
2145 \cs_new_protected:Nn \__stex_symdecl_args:n {
2146   \str_clear:N \l_stex_symdecl_name_str
2147   \str_clear:N \l_stex_symdecl_args_str
2148   \str_clear:N \l_stex_symdecl_deprecate_str
2149   \str_clear:N \l_stex_symdecl_reorder_str
2150   \str_clear:N \l_stex_symdecl_assoctype_str
2151   \bool_set_false:N \l_stex_symdecl_local_bool
2152   \tl_clear:N \l_stex_symdecl_type_tl
2153   \tl_clear:N \l_stex_symdecl_definiens_tl
2154
2155   \keys_set:nn { stex / symdecl } { #1 }
2156 }
```

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

```
2157
2158 \NewDocumentCommand \symdecl { s m O{} } {
2159   \__stex_symdecl_args:n { #3 }
2160   \IfBooleanTF #1 {
2161     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2162   } {
2163     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2164   }
2165   \stex_symdecl_do:n { #2 }
2166   \stex_smsmode_do:
2167 }
```

```

2168
2169 \cs_new_protected:Nn \stex_symdecl_do:nn {
2170   \__stex_symdecl_args:n{#1}
2171   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2172   \stex_symdecl_do:n{#2}
2173 }
2174
2175 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2176 \cs_new_protected:Nn \stex_symdecl_do:n {
2177   \stex_if_in_module:F {
2178     % TODO throw error? some default namespace?
2179   }
2180
2181   \str_if_empty:NT \l_stex_symdecl_name_str {
2182     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2183   }
2184
2185   \prop_if_exist:cT { l_stex_symdecl_
2186     \l_stex_current_module_str ?
2187     \l_stex_symdecl_name_str
2188     _prop
2189   }{
2190     % TODO throw error (beware of circular dependencies)
2191   }
2192
2193   \prop_clear:N \l_tmpa_prop
2194   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2195   \seq_clear:N \l_tmpa_seq
2196   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2197   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2198
2199   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2200     \str_if_empty:NF \l_stex_module_deprecate_str {
2201       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2202     }
2203   }
2204   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2205
2206   \exp_args:No \stex_add_constant_to_current_module:n {
2207     \l_stex_symdecl_name_str
2208   }
2209
2210   % arity/args
2211   \int_zero:N \l_tmpb_int
2212
2213   \bool_set_true:N \l_tmpa_bool
2214   \str_map_inline:Nn \l_stex_symdecl_args_str {
2215     \token_case_meaning:NnF ##1 {
2216       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2217       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2218     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2219     {\tl_to_str:n a} {
2220       \bool_set_false:N \l_tmpa_bool
2221       \int_incr:N \l_tmpb_int
2222     }
2223     {\tl_to_str:n B} {
2224       \bool_set_false:N \l_tmpa_bool
2225       \int_incr:N \l_tmpb_int
2226     }
2227   }{
2228     \msg_error:nnxx{stex}{error/wrongargs}{
2229       \l_stex_current_module_str ?
2230       \l_stex_symdecl_name_str
2231     }{##1}
2232   }
2233 }
2234 \bool_if:NTF \l_tmpa_bool {
2235   % possibly numeric
2236   \str_if_empty:NTF \l_stex_symdecl_args_str {
2237     \prop_put:Nnn \l_tmpa_prop { args } {}
2238     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2239   }{
2240     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2241     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2242     \str_clear:N \l_tmpa_str
2243     \int_step_inline:nn \l_tmpa_int {
2244       \str_put_right:Nn \l_tmpa_str i
2245     }
2246     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2247   }
2248 } {
2249   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2250   \prop_put:Nnx \l_tmpa_prop { arity }
2251   { \str_count:N \l_stex_symdecl_args_str }
2252 }
2253 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2254
2255 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2256   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2257 }{
2258   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2259 }
2260
2261 % semantic macro
2262
2263 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2264   \exp_args:Nx \stex_do_up_to_module:n {
2265     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2266       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2267     }}
2268   }
2269 }
2270
2271 \stex_debug:nn{symbols}{New~symbol:~

```

```

2272 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2273 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2274 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2275 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2276 }
2277
2278 % circular dependencies require this:
2279 \stex_if_do_html:T {
2280   \stex_annotate_invisible:nnn {symdecl} {
2281     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2282   } {
2283     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2284       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2285     }
2286     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$}
2287     \prop_item:Nn \l_tmpa_prop { args }
2288   }
2289   \stex_annotate_invisible:nnn{macroname}{#1}{}
2290   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2291     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2292   }
2293   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2294     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2295   }
2296   \str_if_empty:NF \l_stex_symdecl_reorder_str {
2297     \stex_annotate_invisible:nnn{reorder_args}{\l_stex_symdecl_reorder_str}{}
2298   }
2299 }
2300 }
2301 }
2302 \prop_if_exist:cF {
2303   \l_stex_symdecl_
2304   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2305   _prop
2306 } {
2307   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2308     \__stex_symdecl_restore_symbol:nnnnnnn
2309       {\l_stex_symdecl_name_str}
2310       { \prop_item:Nn \l_tmpa_prop {args} }
2311       { \prop_item:Nn \l_tmpa_prop {arity} }
2312       { \prop_item:Nn \l_tmpa_prop {assoc} }
2313       { \prop_item:Nn \l_tmpa_prop {defined} }
2314       {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2315       {\l_stex_current_module_str}
2316 }
2317 }
2318 }
2319 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2320   \prop_clear:N \l_tmpa_prop
2321   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2322   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2323   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2324   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2325   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }

```



```

2326 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2327 \tl_if_empty:nF{#6}{
2328   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2329 }
2330 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2331 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2332 }

```

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

`\stex_get_symbol:n`

```

2333 \str_new:N \l_stex_get_symbol_uri_str
2334
2335 \cs_new_protected:Nn \stex_get_symbol:n {
2336   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2337     \tl_set:Nn \l_tmpa_tl { #1 }
2338     \__stex_symdecl_get_symbol_from_cs:
2339   }{
2340     % argument is a string
2341     % is it a command name?
2342     \cs_if_exist:cTF { #1 }{
2343       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2344       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2345       \str_if_empty:NTF \l_tmpa_str {
2346         \exp_args:Nx \cs_if_eq:NNTF {
2347           \tl_head:N \l_tmpa_tl
2348         } \stex_invoke_symbol:n {
2349           \__stex_symdecl_get_symbol_from_cs:
2350         }{
2351           \__stex_symdecl_get_symbol_from_string:n { #1 }
2352         }
2353       } {
2354         \__stex_symdecl_get_symbol_from_string:n { #1 }
2355       }
2356     }{
2357       % argument is not a command name
2358       \__stex_symdecl_get_symbol_from_string:n { #1 }
2359       % \l_stex_all_symbols_seq
2360     }
2361   }
2362   \str_if_eq:eeF {
2363     \prop_item:cn {
2364       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2365     }{ deprecate }
2366   }{
2367     \msg_warning:nxxx{stex}{warning/deprecated}{
2368       Symbol~\l_stex_get_symbol_uri_str
2369     }{
2370       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2371     }
2372   }
2373 }
2374
2375 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```

```

2376 \tl_set:Nn \l_tmpa_tl {
2377   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2378 }
2379 \str_set:Nn \l_tmpa_str { #1 }
2380
2381 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2382
2383 \str_if_in:NnTF \l_tmpa_str ? {
2384   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2385   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2386   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2387 }{
2388   \str_clear:N \l_tmpb_str
2389 }
2390 \str_if_empty:NNTF \l_tmpb_str {
2391   \seq_map_inline:Nn \l_stex_all_modules_seq {
2392     \seq_map_inline:cn{c_stex_module_###1_constants}{
2393       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2394         \seq_map_break:n{\seq_map_break:n{
2395           \tl_set:Nn \l_tmpa_tl {
2396             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2397           }
2398         }}
2399       }
2400     }
2401   }
2402 }{
2403   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2404   \seq_map_inline:Nn \l_stex_all_modules_seq {
2405     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2406       \seq_map_inline:cn{c_stex_module_###1_constants}{
2407         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2408           \seq_map_break:n{\seq_map_break:n{
2409             \tl_set:Nn \l_tmpa_tl {
2410               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2411             }
2412           }}
2413         }
2414       }
2415     }
2416   }
2417 }
2418
2419 \l_tmpa_tl
2420 }
2421
2422 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2423   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2424     { \tl_tail:N \l_tmpa_tl }
2425   \tl_if_single:NNTF \l_tmpa_tl {
2426     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2427       \exp_after:wN \str_set:Nn \exp_after:wN
2428         \l_stex_get_symbol_uri_str \l_tmpa_tl
2429     }{

```

```

2430     % TODO
2431     % tail is not a single group
2432   }
2433 }{
2434   % TODO
2435   % tail is not a single group
2436 }
2437 }

```

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

29.2 Notations

```

2438 <@=stex_notation>

notation arguments:
2439 \keys_define:nn { stex / notation } {
2440   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2441   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2442   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2443   op       .tl_set:N   = \l__stex_notation_op_tl ,
2444   primary  .bool_set:N = \l__stex_notation_primary_bool ,
2445   primary  .default:n  = {true} ,
2446   unknown  .code:n     = \str_set:Nx
2447               \l__stex_notation_variant_str \l_keys_key_str
2448 }
2449
2450 \cs_new_protected:Nn \stex_notation_args:n {
2451   % \str_clear:N \l__stex_notation_lang_str
2452   \str_clear:N \l__stex_notation_variant_str
2453   \str_clear:N \l__stex_notation_prec_str
2454   \tl_clear:N \l__stex_notation_op_tl
2455   \bool_set_false:N \l__stex_notation_primary_bool
2456
2457   \keys_set:nn { stex / notation } { #1 }
2458 }

\notation
2459 \NewDocumentCommand \notation { s m 0{}} {
2460   \stex_notation_args:n { #3 }
2461   \tl_clear:N \l_stex_symdecl_definiens_tl
2462   \stex_get_symbol:n { #2 }
2463   \tl_set:Nn \l_stex_notation_after_do_tl {
2464     \__stex_notation_final:
2465     \IfBooleanTF#1{
2466       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2467     }{}
2468     \stex_smsmode_do:\ignorespacesandpars
2469   }
2470   \stex_notation_do:nnnnn
2471   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2472   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2473   { \l__stex_notation_variant_str }
2474   { \l__stex_notation_prec_str }

```

```

2475 }
2476 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2477 \seq_new:N \l__stex_notation_precedences_seq
2478 \tl_new:N \l__stex_notation_opprec_tl
2479 \int_new:N \l__stex_notation_currarg_int
2480 \tl_new:N \stex_symbol_after_invokation_tl
2481
2482 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2483   \let\l_stex_current_symbol_str\relax
2484   \seq_clear:N \l__stex_notation_precedences_seq
2485   \tl_clear:N \l__stex_notation_opprec_tl
2486   \str_set:Nx \l__stex_notation_args_str { #1 }
2487   \str_set:Nx \l__stex_notation_arity_str { #2 }
2488   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2489   \str_set:Nx \l__stex_notation_prec_str { #4 }
2490
2491   % precedences
2492   \str_if_empty:NTF \l__stex_notation_prec_str {
2493     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2494       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2495     }{
2496       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2497     }
2498   } {
2499     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2500       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2501       \int_step_inline:nn { \l__stex_notation_arity_str } {
2502         \exp_args:NNo
2503         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2504       }
2505     }{
2506       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2507       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2508         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2509         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2510           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2511             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2512           \seq_map_inline:Nn \l_tmpa_seq {
2513             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2514           }
2515         }
2516       }{
2517         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2518           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2519         }{
2520           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2521         }
2522       }
2523     }
2524   }

```

```

2525 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2526 \int_step_inline:nn { \l__stex_notation_arity_str } {
2527   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2528     \exp_args:NNo
2529     \seq_put_right:No \l__stex_notation_precedences_seq {
2530       \l__stex_notation_opprec_tl
2531     }
2532   }
2533 }
2534 }
2535 \tl_clear:N \l_stex_notation_dummyargs_tl
2536
2537 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2538   \exp_args:NNe
2539   \cs_set:Npn \l_stex_notation_macrocode_cs {
2540     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2541     { \l__stex_notation_suffix_str }
2542     { \l__stex_notation_opprec_tl }
2543     { \exp_not:n { #5 } }
2544   }
2545   \l_stex_notation_after_do_tl
2546 }{
2547   \str_if_in:NnTF \l__stex_notation_args_str b {
2548     \exp_args:Nne \use:nn
2549     {
2550       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2551       \cs_set:Npn \l__stex_notation_arity_str } { {
2552         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2553         { \l__stex_notation_suffix_str }
2554         { \l__stex_notation_opprec_tl }
2555         { \exp_not:n { #5 } }
2556       } }
2557 }{
2558   \str_if_in:NnTF \l__stex_notation_args_str B {
2559     \exp_args:Nne \use:nn
2560     {
2561       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2562       \cs_set:Npn \l__stex_notation_arity_str } { {
2563         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2564         { \l__stex_notation_suffix_str }
2565         { \l__stex_notation_opprec_tl }
2566         { \exp_not:n { #5 } }
2567       } }
2568 }{
2569   \exp_args:Nne \use:nn
2570   {
2571     \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2572     \cs_set:Npn \l__stex_notation_arity_str } { {
2573       \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2574       { \l__stex_notation_suffix_str }
2575       { \l__stex_notation_opprec_tl }
2576       { \exp_not:n { #5 } }
2577     } }
2578 }

```

```

2579     }
2580
2581     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2582     \int_zero:N \l__stex_notation_currarg_int
2583     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2584     \__stex_notation_arguments:
2585   }
2586 }

```

(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

```

2587 \cs_new_protected:Nn \__stex_notation_arguments: {
2588   \int_incr:N \l__stex_notation_currarg_int
2589   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2590     \l_stex_notation_after_do_tl
2591   }{
2592     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2593     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2594     \str_if_eq:VnTF \l_tmpa_str a {
2595       \__stex_notation_argument_assoc:nn{a}
2596     }{
2597       \str_if_eq:VnTF \l_tmpa_str B {
2598         \__stex_notation_argument_assoc:nn{B}
2599       }{
2600         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2601         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2602           { \stex_term_math_arg:nnn
2603             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2604             { \l_tmpb_str }
2605             { ###\int_use:N \l__stex_notation_currarg_int }
2606           }
2607         }
2608         \__stex_notation_arguments:
2609       }
2610     }
2611   }
2612 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2613 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2614
2615   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2616     {\l__stex_notation_arity_str}{
2617     #2
2618   }
2619   \int_zero:N \l_tmpa_int
2620   \tl_clear:N \l_tmpa_tl
2621   \str_map_inline:Nn \l__stex_notation_args_str {
2622     \int_incr:N \l_tmpa_int
2623     \tl_put_right:Nx \l_tmpa_tl {
2624       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```

```

2625     \str_if_eq:nnTF {##1}{B}{ } }{
2626     { \_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2627     }
2628     }
2629     }
2630     }
2631     \exp_after:wN\exp_after:wN\exp_after:wN \def
2632     \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2633     \exp_after:wN\exp_after:wN\exp_after:wN ##
2634     \exp_after:wN\exp_after:wN\exp_after:wN 1
2635     \exp_after:wN\exp_after:wN\exp_after:wN ##
2636     \exp_after:wN\exp_after:wN\exp_after:wN 2
2637     \exp_after:wN\exp_after:wN\exp_after:wN {
2638     \exp_after:wN \exp_after:wN \exp_after:wN
2639     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2640     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2641     }
2642     }
2643
2644     \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2645     \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2646     \_stex_term_math_assoc_arg:nnnn
2647     { #1\int_use:N \l__stex_notation_currarg_int }
2648     { \l_tmpa_str }
2649     { #####\int_use:N \l__stex_notation_currarg_int }
2650     { \l_tmpa_cs {####1} {####2} }
2651     } }
2652     \__stex_notation_arguments:
2653     }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2654 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2655     \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2656     \cs_set_nopar:Npn {#3}{#4}
2657     \tl_if_empty:nF {#5}{
2658     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{ \comp{ #5 }
2659     }
2660     \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2661     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2662     }
2663     }
2664
2665     \cs_new_protected:Nn \_stex_notation_final: {
2666
2667     \stex_execute_in_module:x {
2668     \_stex_notation_restore_notation:nnnnn
2669     {\l_stex_get_symbol_uri_str}
2670     {\l_stex_notation_suffix_str}
2671     {\l__stex_notation_arity_str}
2672     {
2673     \exp_after:wN \exp_after:wN \exp_after:wN
2674     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2675     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2676   }
2677   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2678 }
2679
2680 \stex_debug:nn{symbols}{
2681   Notation~\l__stex_notation_suffix_str
2682   ~for~\l_stex_get_symbol_uri_str^^J
2683   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2684   Argument~precedences:~
2685     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2686   Notation: \cs_meaning:c {
2687     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2688     \l__stex_notation_suffix_str
2689     _cs
2690   }
2691 }
2692 % HTML annotations
2693 \stex_if_do_html:T {
2694   \stex_annotate_invisible:nnn { notation }
2695   { \l_stex_get_symbol_uri_str } {
2696     \stex_annotate_invisible:nnn { notationfragment }
2697     { \l__stex_notation_suffix_str }{}
2698     \stex_annotate_invisible:nnn { precedence }
2699     { \l__stex_notation_prec_str }{}
2700
2701     \int_zero:N \l_tmpa_int
2702     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2703     \tl_clear:N \l_tmpa_tl
2704     \int_step_inline:nn { \l__stex_notation_arity_str }{
2705       \int_incr:N \l_tmpa_int
2706       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2707       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2708       \str_if_eq:VnTF \l_tmpb_str a {
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         \str_if_eq:VnTF \l_tmpb_str B {
2715           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2716             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2717             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2718           } }
2719         }{
2720           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2721             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2722           } }
2723         }
2724       }
2725     }
2726     \stex_annotate_invisible:nnn { notationcomp }{}{
2727       \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2728       $ \exp_args:Nno \use:nn { \use:c {

```



```

2729         stex_notation_ \l_stex_current_symbol_str
2730         \c_hash_str \l__stex_notation_suffix_str _cs
2731     } } { \l_tmpa_tl } $
2732   }
2733 }
2734 }
2735 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2736 \keys_define:nn { stex / setnotation } {
2737   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2738   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2739   unknown .code:n      = \str_set:Nx
2740     \l__stex_notation_variant_str \l_keys_key_str
2741 }
2742
2743 \cs_new_protected:Nn \stex_setnotation_args:n {
2744   % \str_clear:N \l__stex_notation_lang_str
2745   \str_clear:N \l__stex_notation_variant_str
2746   \keys_set:nn { stex / setnotation } { #1 }
2747 }
2748
2749 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2750   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2751     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2752     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2753   }
2754 }
2755
2756 \cs_new_protected:Nn \stex_setnotation:n {
2757   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2758     { \l__stex_notation_variant_str }{
2759     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2760     \stex_debug:nn {notations}{
2761       Setting~default~notation~
2762       {\l__stex_notation_variant_str }~for~
2763       #1 \\
2764       \expandafter\meaning\csname
2765       l_stex_symdecl_#1_notations\endcsname
2766     }
2767   }{
2768     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2769   }
2770 }
2771
2772 \NewDocumentCommand \setnotation {m m} {
2773   \stex_get_symbol:n { #1 }
2774   \stex_setnotation_args:n { #2 }
2775   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2776   \stex_smsmode_do:\ignorespacesandpars
2777 }
2778

```

```

2779 \cs_new_protected:Nn \stex_copy_notations:nn {
2780   \stex_debug:nn {notations}{
2781     Copying~notations~from~#2~to~#1\\
2782     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2783   }
2784   \tl_clear:N \l_tmpa_tl
2785   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2786     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }
2787   }
2788   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2789     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2790     \edef \l_tmpa_tl {
2791       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2792       \exp_after:wN\exp_after:wN\exp_after:wN {
2793         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2794       }
2795     }
2796
2797     \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2798     \exp_after:wN #####\exp_after:wN 1 \exp_after:wN #####\exp_after:wN 2
2799     \exp_after:wN { \l_tmpa_tl }
2800
2801     \edef \l_tmpa_tl {
2802       \exp_after:wN \exp_not:n \exp_after:wN {
2803         \l_tmpa_tl {##### 1}{##### 2}
2804       }
2805     }
2806
2807     \stex_execute_in_module:x {
2808       \__stex_notation_restore_notation:nnnnn
2809       {#1}{##1}
2810       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2811       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2812       {
2813         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2814           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2815             }
2816         }
2817       }
2818     }
2819   }
2820
2821   \NewDocumentCommand \copynotation {m m} {
2822     \stex_get_symbol:n { #1 }
2823     \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2824     \stex_get_symbol:n { #2 }
2825     \exp_args:Noo
2826     \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2827     \stex_smsmode_do:\ignorespacesandpars
2828   }
2829

```

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

\symdef

```
2830 \keys_define:nn { stex / symdef } {
2831   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2832   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2833   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2834   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2835   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2836   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2837   op        .tl_set:N    = \l__stex_notation_op_tl ,
2838   % lang     .str_set_x:N = \l__stex_notation_lang_str ,
2839   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2840   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2841   assoc     .choices:nn =
2842     {bin,binl,binr,pre,conj,pwconj}
2843     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},
2844   unknown   .code:n      = \str_set:Nx
2845     \l__stex_notation_variant_str \l_keys_key_str
2846 }
2847
2848 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2849   \str_clear:N \l_stex_symdecl_name_str
2850   \str_clear:N \l_stex_symdecl_args_str
2851   \str_clear:N \l_stex_symdecl_assoc_type_str
2852   \str_clear:N \l_stex_symdecl_reorder_str
2853   \bool_set_false:N \l_stex_symdecl_local_bool
2854   \tl_clear:N \l_stex_symdecl_type_tl
2855   \tl_clear:N \l_stex_symdecl_definiens_tl
2856   % \str_clear:N \l__stex_notation_lang_str
2857   \str_clear:N \l__stex_notation_variant_str
2858   \str_clear:N \l__stex_notation_prec_str
2859   \tl_clear:N \l__stex_notation_op_tl
2860
2861   \keys_set:nn { stex / symdef } { #1 }
2862 }
2863
2864 \NewDocumentCommand \symdef { m O{} } {
2865   \__stex_notation_symdef_args:n { #2 }
2866   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2867   \stex_symdecl_do:n { #1 }
2868   \tl_set:Nn \l_stex_notation_after_do_tl {
2869     \__stex_notation_final:
2870     \stex_smsmode_do:\ignorespacesandpars
2871   }
2872   \str_set:Nx \l_stex_get_symbol_uri_str {
2873     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2874   }
2875   \exp_args:Nx \stex_notation_do:nnnnn
2876     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2877     { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2878     { \l__stex_notation_variant_str }
2879     { \l__stex_notation_prec_str }
2880 }
2881 \stex_deactivate_macro:Nn \symdef {module~environments}
```

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

29.3 Variables

```

2882 <@@=stex_variables>
2883
2884 \keys_define:nn { stex / vardef } {
2885   name      .str_set_x:N = \l__stex_variables_name_str ,
2886   args      .str_set_x:N = \l__stex_variables_args_str ,
2887   type      .tl_set:N    = \l__stex_variables_type_tl ,
2888   def       .tl_set:N    = \l__stex_variables_def_tl ,
2889   op        .tl_set:N    = \l__stex_variables_op_tl ,
2890   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2891   assoc     .choices:nn =
2892     {bin,binl,binr,pre,conj,pwconj}
2893     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2894   bind      .choices:nn =
2895     {forall,exists}
2896     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2897 }
2898
2899 \cs_new_protected:Nn \__stex_variables_args:n {
2900   \str_clear:N \l__stex_variables_name_str
2901   \str_clear:N \l__stex_variables_args_str
2902   \str_clear:N \l__stex_variables_prec_str
2903   \str_clear:N \l__stex_variables_assoctype_str
2904   \str_clear:N \l__stex_variables_bind_str
2905   \tl_clear:N \l__stex_variables_type_tl
2906   \tl_clear:N \l__stex_variables_def_tl
2907   \tl_clear:N \l__stex_variables_op_tl
2908
2909   \keys_set:nn { stex / vardef } { #1 }
2910 }
2911
2912 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2913   \__stex_variables_args:n {#2}
2914   \str_if_empty:NT \l__stex_variables_name_str {
2915     \str_set:Nx \l__stex_variables_name_str { #1 }
2916   }
2917   \prop_clear:N \l_tmpa_prop
2918   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2919
2920   \int_zero:N \l_tmpb_int
2921   \bool_set_true:N \l_tmpa_bool
2922   \str_map_inline:Nn \l__stex_variables_args_str {
2923     \token_case_meaning:NnF ##1 {
2924       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2925       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2926       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2927       {\tl_to_str:n a} {
2928         \bool_set_false:N \l_tmpa_bool
2929         \int_incr:N \l_tmpb_int
2930       }

```

```

2931     {\tl_to_str:n B} {
2932         \bool_set_false:N \l_tmpa_bool
2933         \int_incr:N \l_tmpb_int
2934     }
2935 }{
2936     \msg_error:nxxx{stex}{error/wrongargs}{
2937         variable~\l__stex_variables_name_str
2938     }{##1}
2939 }
2940 }
2941 \bool_if:NTF \l_tmpa_bool {
2942     % possibly numeric
2943     \str_if_empty:NTF \l__stex_variables_args_str {
2944         \prop_put:Nnn \l_tmpa_prop { args } {}
2945         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2946     }{
2947         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2948         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2949         \str_clear:N \l_tmpa_str
2950         \int_step_inline:nn \l_tmpa_int {
2951             \str_put_right:Nn \l_tmpa_str i
2952         }
2953         \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2954         \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2955     }
2956 } {
2957     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2958     \prop_put:Nnx \l_tmpa_prop { arity }
2959     { \str_count:N \l__stex_variables_args_str }
2960 }
2961 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2962 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2963
2964 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2965
2966 \tl_if_empty:NF \l__stex_variables_op_tl {
2967     \cs_set:cpx {
2968         stex_var_op_notation_\l__stex_variables_name_str_cs
2969     } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2970 }
2971
2972 \tl_set:Nn \l_stex_notation_after_do_tl {
2973     \exp_args:Nne \use:nn {
2974         \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2975         \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2976     } {{
2977         \exp_after:wN \exp_after:wN \exp_after:wN
2978         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2979         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2980     }}
2981     \stex_if_do_html:T {
2982         \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2983             \stex_annotate_invisible:nnn { precedence }
2984             { \l__stex_variables_prec_str }{}

```

```

2985 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{}}{\$ \l
2986 \stex_annotate_invisible:nnn{args}{}}{\l__stex_variables_args_str }
2987 \stex_annotate_invisible:nnn{macroname}{#1}{ }
2988 \tl_if_empty:NF \l__stex_variables_def_tl {
2989 \stex_annotate_invisible:nnn{definiens}{ }
2990 {\$ \l__stex_variables_def_tl$}
2991 }
2992 \str_if_empty:NF \l__stex_variables_assoctype_str {
2993 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{ }
2994 }
2995 \str_if_empty:NF \l__stex_variables_bind_str {
2996 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{ }
2997 }
2998 \int_zero:N \l_tmpa_int
2999 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3000 \tl_clear:N \l_tmpa_tl
3001 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3002 \int_incr:N \l_tmpa_int
3003 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3004 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3005 \str_if_eq:VnTF \l_tmpb_str a {
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 \str_if_eq:VnTF \l_tmpb_str B {
3012 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3013 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
3014 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
3015 } }
3016 }{
3017 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3018 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
3019 } }
3020 }
3021 }
3022 }
3023 \stex_annotate_invisible:nnn { notationcomp }{ }{
3024 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
3025 $ \exp_args:Nno \use:nn { \use:c {
3026 stex_var_notation_\l__stex_variables_name_str _cs
3027 } } { \l_tmpa_tl } $
3028 }
3029 }
3030 }\ignorespacesandpars
3031 }
3032
3033 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3034 }
3035
3036 \cs_new:Nn \_stex_reset:N {
3037 \tl_if_exist:NTF #1 {
3038 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }

```

```

3039 }{
3040   \let \exp_not:N #1 \exp_not:N \undefined
3041 }
3042 }
3043
3044 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3045   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3046   \exp_args:Nnx \use:nn {
3047     % TODO
3048     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3049       #2
3050     }
3051   }{
3052     \stex_reset:N \varnot
3053     \stex_reset:N \vartype
3054     \stex_reset:N \vardefi
3055   }
3056 }
3057
3058 \NewDocumentCommand \vardef { s } {
3059   \IfBooleanTF#1 {
3060     \__stex_variables_do_complex:nn
3061   }{
3062     \__stex_variables_do_simple:nnn
3063   }
3064 }
3065
3066 \NewDocumentCommand \svar { 0{} m }{
3067   \tl_if_empty:nTF {#1}{
3068     \str_set:Nn \l_tmpa_str { #2 }
3069   }{
3070     \str_set:Nn \l_tmpa_str { #1 }
3071   }
3072   \stex_term_omv:nn {
3073     var://\l_tmpa_str
3074   }{
3075     \exp_args:Nnx \use:nn {
3076       \def\comp{\_varcomp}
3077       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3078       \comp{ #2 }
3079     }{
3080       \stex_reset:N \comp
3081       \stex_reset:N \l_stex_current_symbol_str
3082     }
3083   }
3084 }
3085
3086
3087
3088 \keys_define:nn { stex / varseq } {
3089   name .str_set_x:N = \l__stex_variables_name_str ,
3090   args .int_set:N   = \l__stex_variables_args_int ,
3091   type .tl_set:N    = \l__stex_variables_type_tl ,
3092   mid .tl_set:N     = \l__stex_variables_mid_tl ,

```

```

3093   bind      .choices:nn      =
3094   {forall,exists}
3095   {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3096 }
3097
3098 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3099   \str_clear:N \l__stex_variables_name_str
3100   \int_set:Nn \l__stex_variables_args_int 1
3101   \tl_clear:N \l__stex_variables_type_tl
3102   \str_clear:N \l__stex_variables_bind_str
3103
3104   \keys_set:nn { stex / varseq } { #1 }
3105 }
3106
3107 \NewDocumentCommand \varseq {m O{}} m m m){
3108   \__stex_variables_seq_args:n { #2 }
3109   \str_if_empty:NT \l__stex_variables_name_str {
3110     \str_set:Nx \l__stex_variables_name_str { #1 }
3111   }
3112   \prop_clear:N \l_tmpa_prop
3113   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3114
3115   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3116   \int_compare:nNnF {\seq_count:N \l_tmpa_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_tmpa_seq}
3120   }
3121   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3122   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3123     \msg_error:nnxx{stex}{error/seqlength}
3124     {\int_use:N \l__stex_variables_args_int}
3125     {\seq_count:N \l_tmpb_seq}
3126   }
3127   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3128   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3129
3130   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3131   \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
3132
3133   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3134   \int_step_inline:nn \l__stex_variables_args_int {
3135     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3136   }
3137   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3138   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3139   \tl_if_empty:NF \l__stex_variables_mid_tl {
3140     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3141     \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3142   }
3143   \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3144   \int_step_inline:nn \l__stex_variables_args_int {
3145     \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3146   }

```



```

3147 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3148 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3149
3150
3151 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3152
3153 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3154
3155 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3156
3157 \int_step_inline:nn \l__stex_variables_args_int {
3158   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3159     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3160   }}
3161 }
3162
3163 \tl_set:Nx \l_tmpa_tl {
3164   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3165     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3166   }
3167 }
3168
3169 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3170
3171 \exp_args:Nno \use:nn {
3172   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3173   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3174
3175   \stex_debug:nn{sequences}{New~Sequence:~
3176     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str _cs\endcsname\\~\\
3177     \prop_to_keyval:N \l_tmpa_prop
3178   }
3179   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3180     \tl_if_empty:NF \l__stex_variables_type_tl {
3181       \stex_annotate:nnn {type}{\}{\seqtype\l__stex_variables_type_tl$}
3182     }
3183     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3184     \str_if_empty:NF \l__stex_variables_bind_str {
3185       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3186     }
3187   }}
3188
3189   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str _prop}\l_tmpa_prop
3190   \ignorespacesandpars
3191 }
3192
3193 </package>

```

Chapter 30

STEX -Terms Implementation

```
3194 <*package>
3195
3196 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3197
3198 <@@=stex_terms>
3199
3200 Warnings and error messages
3201 \msg_new:nnn{stex}{error/nonotation}{
3202   Symbol~#1~invoked,~but~has~no~notation~#2!
3203 }
3204 \msg_new:nnn{stex}{error/notationarg}{
3205   Error~in~parsing~notation~#1
3206 }
3207 \msg_new:nnn{stex}{error/noop}{
3208   Symbol~#1~has~no~operator~notation~for~notation~#2
3209 }
3210 \msg_new:nnn{stex}{error/notallowed}{
3211   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3212 }
3213 \msg_new:nnn{stex}{error/doubleargument}{
3214   Argument~#1~of~symbol~#2~already~assigned
3215 }
3216 \msg_new:nnn{stex}{error/overarity}{
3217   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3218 }
3219
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3218
3219
3220 \bool_new:N \l_stex_allow_semantic_bool
3221 \bool_set_true:N \l_stex_allow_semantic_bool
3222
```

```

3223 \cs_new_protected:Nn \stex_invoke_symbol:n {
3224   \bool_if:NTF \l_stex_allow_semantic_bool {
3225     \str_if_eq:eeF {
3226       \prop_item:cn {
3227         l_stex_symdecl_#1_prop
3228       }{ deprecate }
3229     }{}{
3230       \msg_warning:nxxx{stex}{warning/deprecated}{
3231         Symbol~#1
3232       }{
3233         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3234       }
3235     }
3236     \if_mode_math:
3237       \exp_after:wN \__stex_terms_invoke_math:n
3238     \else:
3239       \exp_after:wN \__stex_terms_invoke_text:n
3240     \fi: { #1 }
3241   }{
3242     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3243   }
3244 }
3245
3246 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3247   \peek_charcode_remove:NTF ! {
3248     \__stex_terms_invoke_op_custom:nn {#1}
3249   }{
3250     \__stex_terms_invoke_custom:nn {#1}
3251   }
3252 }
3253
3254 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3255   \peek_charcode_remove:NTF ! {
3256     % operator
3257     \peek_charcode_remove:NTF * {
3258       % custom op
3259       \__stex_terms_invoke_op_custom:nn {#1}
3260     }{
3261       % op notation
3262       \peek_charcode:NTF [ {
3263         \__stex_terms_invoke_op_notation:nw {#1}
3264       }{
3265         \__stex_terms_invoke_op_notation:nw {#1}[]
3266       }
3267     }
3268   }{
3269     \peek_charcode_remove:NTF * {
3270       \__stex_terms_invoke_custom:nn {#1}
3271       % custom
3272     }{
3273       % normal
3274       \peek_charcode:NTF [ {
3275         \__stex_terms_invoke_notation:nw {#1}
3276       }{

```

```

3277     \__stex_terms_invoke_notation:nw {#1}[]
3278   }
3279 }
3280 }
3281 }
3282
3283
3284 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3285   \exp_args:Nnx \use:nn {
3286     \def\comp{\_comp}
3287     \str_set:Nn \l_stex_current_symbol_str { #1 }
3288     \bool_set_false:N \l_stex_allow_semantic_bool
3289     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3290       \comp{ #2 }
3291     }
3292   }{
3293     \stex_reset:N \comp
3294     \stex_reset:N \l_stex_current_symbol_str
3295     \bool_set_true:N \l_stex_allow_semantic_bool
3296   }
3297 }
3298
3299 \keys_define:nn { stex / terms } {
3300   % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3301   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3302   unknown .code:n = \str_set:Nx
3303     \l_stex_notation_variant_str \l_keys_key_str
3304 }
3305
3306 \cs_new_protected:Nn \__stex_terms_args:n {
3307   % \str_clear:N \l_stex_notation_lang_str
3308   \str_clear:N \l_stex_notation_variant_str
3309
3310   \keys_set:nn { stex / terms } { #1 }
3311 }
3312
3313 \cs_new_protected:Nn \stex_find_notation:nn {
3314   \__stex_terms_args:n { #2 }
3315   \seq_if_empty:cTF {
3316     l_stex_symdecl_ #1 _notations
3317   } {
3318     \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3319   } {
3320     \str_if_empty:NTF \l_stex_notation_variant_str {
3321       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3322     }{
3323       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3324         \l_stex_notation_variant_str
3325       }{
3326         % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3327       }{
3328         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3329           ~\l_stex_notation_variant_str
3330         }
3331       }
3332     }
3333   }

```

```

3331     }
3332   }
3333 }
3334 }
3335
3336 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3337   \exp_args:Nnx \use:nn {
3338     \def\comp{\_comp}
3339     \str_set:Nn \l_stex_current_symbol_str { #1 }
3340     \stex_find_notation:nn { #1 }{ #2 }
3341     \bool_set_false:N \l_stex_allow_semantic_bool
3342     \cs_if_exist:cTF {
3343       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3344     }{
3345       \_stex_term_oms:nnn { #1 }{
3346         #1 \c_hash_str \l_stex_notation_variant_str
3347       }{
3348         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3349       }
3350     }{
3351       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3352         \cs_if_exist:cTF {
3353           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3354         }{
3355           \tl_set:Nx \stex_symbol_after_invokation_tl {
3356             \_stex_reset:N \comp
3357             \_stex_reset:N \stex_symbol_after_invokation_tl
3358             \_stex_reset:N \l_stex_current_symbol_str
3359             \bool_set_true:N \l_stex_allow_semantic_bool
3360           }
3361           \def\comp{\_comp}
3362           \str_set:Nn \l_stex_current_symbol_str { #1 }
3363           \bool_set_false:N \l_stex_allow_semantic_bool
3364           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3365         }{
3366           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3367             ~\l_stex_notation_variant_str
3368           }
3369         }
3370       }{
3371         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3372       }
3373     }
3374   }{
3375     \_stex_reset:N \comp
3376     \_stex_reset:N \l_stex_current_symbol_str
3377     \bool_set_true:N \l_stex_allow_semantic_bool
3378   }
3379 }
3380
3381 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3382   \stex_find_notation:nn { #1 }{ #2 }
3383   \cs_if_exist:cTF {
3384     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3385 }{
3386   \tl_set:Nx \stex_symbol_after_invokation_tl {
3387     \_stex_reset:N \comp
3388     \_stex_reset:N \stex_symbol_after_invokation_tl
3389     \_stex_reset:N \l_stex_current_symbol_str
3390     \bool_set_true:N \l_stex_allow_semantic_bool
3391   }
3392   \def\comp{\_comp}
3393   \str_set:Nn \l_stex_current_symbol_str { #1 }
3394   \bool_set_false:N \l_stex_allow_semantic_bool
3395   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3396 }{
3397   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3398     ~\l_stex_notation_variant_str
3399   }
3400 }
3401 }
3402
3403 \prop_new:N \l__stex_terms_custom_args_prop
3404
3405 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3406   \exp_args:Nnx \use:nn {
3407     \bool_set_false:N \l_stex_allow_semantic_bool
3408     \def\comp{\_comp}
3409     \str_set:Nn \l_stex_current_symbol_str { #1 }
3410     \prop_clear:N \l__stex_terms_custom_args_prop
3411     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3412     \prop_get:cnN {
3413       l_stex_symdecl_#1 _prop
3414     }{ args } \l_tmpa_str
3415     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3416     \tl_set:Nn \arg { \__stex_terms_arg: }
3417     \str_if_empty:NTF \l_tmpa_str {
3418       \_stex_term oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3419     }{
3420       \str_if_in:NnTF \l_tmpa_str b {
3421         \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3422       }{
3423         \str_if_in:NnTF \l_tmpa_str B {
3424           \_stex_term ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3425         }{
3426           \_stex_term oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3427         }
3428       }
3429     }
3430     % TODO check that all arguments exist
3431   }{
3432     \_stex_reset:N \l_stex_current_symbol_str
3433     \_stex_reset:N \arg
3434     \_stex_reset:N \comp
3435     \_stex_reset:N \l__stex_terms_custom_args_prop
3436     \bool_set_true:N \l_stex_allow_semantic_bool
3437   }
3438 }

```

```

3439
3440 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3441   \tl_if_empty:nTF {#2}{
3442     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3443     \bool_set_true:N \l_tmpa_bool
3444     \bool_do_while:Nn \l_tmpa_bool {
3445       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3446         \int_incr:N \l_tmpa_int
3447       }{
3448         \bool_set_false:N \l_tmpa_bool
3449       }
3450     }
3451   }{
3452     \int_set:Nn \l_tmpa_int { #2 }
3453   }
3454   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3455   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3456     \msg_error:nnxxx{stex}{error/overarity}
3457     {\int_use:N \l_tmpa_int}
3458     {\l_stex_current_symbol_str}
3459     {\str_count:N \l_tmpa_str}
3460   }
3461   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3462   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3463     \bool_lazy_any:nF {
3464       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3465       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3466     }{
3467       \msg_error:nnxx{stex}{error/doubleargument}
3468       {\int_use:N \l_tmpa_int}
3469       {\l_stex_current_symbol_str}
3470     }
3471   }
3472   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3473   \bool_set_true:N \l_stex_allow_semantic_bool
3474   \IfBooleanTF#1{
3475     \stex_annotate_invisible:n { %TODO
3476       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3477     }
3478   }{ %TODO
3479     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3480   }
3481   \bool_set_false:N \l_stex_allow_semantic_bool
3482 }
3483
3484
3485 \cs_new_protected:Nn \_stex_term_arg:nn {
3486   \bool_set_true:N \l_stex_allow_semantic_bool
3487   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3488   \bool_set_false:N \l_stex_allow_semantic_bool
3489 }
3490
3491 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3492   \exp_args:Nnx \use:nn

```

```

3493 { \int_set:Nn \l__stex_terms_downprec { #2 }
3494   \stex_term_arg:nn { #1 }{ #3 }
3495 }
3496 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3497 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

3498 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3499   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3500   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3501   \tl_if_empty:NTF { #3 }{
3502     \stex_term_math_arg:nnn{#1}{#2}{#3}
3503   }{
3504     \exp_args:Nx \tl_if_empty:NTF { \tl_tail:n{ #3 } }{
3505       \expandafter\if\expandafter\relax\noexpand#3
3506       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3507     }else
3508       \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3509     \fi
3510     \l_tmpa_tl
3511   }{
3512     \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3513   }
3514 }
3515 }
3516
3517 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3518   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3519   \str_if_empty:NTF \l_tmpa_str {
3520     \exp_args:Nx \cs_if_eq:NNTF {
3521       \tl_head:N #1
3522     } \stex_invoke_sequence:n {
3523       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3524       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3525       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3526       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3527       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3528         \exp_not:n{\exp_args:Nnx \use:nn} {
3529           \exp_not:n {
3530             \def\comp{\_varcomp}
3531             \str_set:Nn \l_stex_current_symbol_str
3532             } {varseq://\l_tmpa_str}
3533           \exp_not:n{ ##1 }
3534         }{
3535           \exp_not:n {
3536             \stex_reset:N \comp
3537             \stex_reset:N \l_stex_current_symbol_str
3538           }
3539         }
3540       }}}
3541       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3542       \seq_reverse:N \l_tmpa_seq

```



```

3543 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3544 \seq_map_inline:Nn \l_tmpa_seq {
3545   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3546     \exp_args:Nno
3547     \l_tmpa_cs { ##1 } \l_tmpa_tl
3548   }
3549 }
3550 \tl_set:Nx \l_tmpa_tl {
3551   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3552     \exp_args:No \exp_not:n \l_tmpa_tl
3553   }
3554 }
3555 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3556 }{
3557   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3558 }
3559 } {
3560   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3561 }
3562 }
3563 }
3564
3565 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3566   \clist_set:Nn \l_tmpa_clist{ #2 }
3567   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3568     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3569   }{
3570     \clist_reverse:N \l_tmpa_clist
3571     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3572     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3573       \exp_args:No \exp_not:n \l_tmpa_tl
3574     }}
3575     \clist_map_inline:Nn \l_tmpa_clist {
3576       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3577         \exp_args:Nno
3578         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3579       }
3580     }
3581   }
3582   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3583 }

```

(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
3584 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3585 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3586 \int_new:N \l__stex_terms_downprec
3587 \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`

```
3588 \tl_set:Nn \l__stex_terms_left_bracket_str (
3589 \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

```
3590 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3591   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3592     \bool_set_false:N \l__stex_terms_brackets_done_bool
3593     #2
3594   } {
3595     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3596       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3597         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3598         \dobrackets { #2 }
3599       }
3600     }{ #2 }
3601   }
3602 }
```

(End definition for `__stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
3603 \bool_new:N \l__stex_terms_brackets_done_bool
3604 %\RequirePackage{scalerel}
3605 \cs_new_protected:Npn \dobrackets #1 {
3606   %\ThisStyle{\if D\m@switch
3607   %   \exp_args:Nnx \use:nn
3608   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3609   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3610   %   \else
3611   \exp_args:Nnx \use:nn
3612   {
3613     \bool_set_true:N \l__stex_terms_brackets_done_bool
3614     \int_set:Nn \l__stex_terms_downprec \infprec
3615     \l__stex_terms_left_bracket_str
3616     #1
3617   }
3618   {
3619     \bool_set_false:N \l__stex_terms_brackets_done_bool
3620     \l__stex_terms_right_bracket_str
3621     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3622   }
3623   %\fi}
3624 }
```

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

\withbrackets

```
3625 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3626   \exp_args:Nnx \use:nn
3627   {
3628     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3629     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3630     #3
3631   }
3632   {
3633     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3634     {\l__stex_terms_left_bracket_str}
3635     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3636     {\l__stex_terms_right_bracket_str}
3637   }
3638 }
```

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

\STEXinvisible

```
3639 \cs_new_protected:Npn \STEXinvisible #1 {
3640   \stex_annotate_invisible:n { #1 }
3641 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
3642 \cs_new_protected:Nn \_stex_term_oms:nnn {
3643   \stex_annotate:nnn{ OMID }{ #2 }{
3644     #3
3645   }
3646 }
3647
3648 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3649   \__stex_terms_maybe_brackets:nn { #3 }{
3650     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3651   }
3652 }
```

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

_stex_term_math_omv:nn

```
3653 \cs_new_protected:Nn \_stex_term_omv:nn {
3654   \stex_annotate:nnn{ OMV }{ #1 }{
3655     #2
3656   }
3657 }
```

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

_stex_term_math_oma:nnnn

```
3658 \cs_new_protected:Nn \_stex_term_oma:nnn {
3659   \stex_annotate:nnn{ OMA }{ #2 }{
3660     #3
3661   }
```

```

3662 }
3663
3664 \cs_new_protected:Nn \stex_term_math_oma:nnnn {
3665   \__stex_terms_maybe_brackets:nn { #3 }{
3666     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3667   }
3668 }

```

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

`\stex_term_math_omb:nnnn`

```

3669 \cs_new_protected:Nn \stex_term_ombind:nnn {
3670   \stex_annotate:nnn{ OMBIND }{ #2 }{
3671     #3
3672   }
3673 }
3674
3675 \cs_new_protected:Nn \stex_term_math_omb:nnnn {
3676   \__stex_terms_maybe_brackets:nn { #3 }{
3677     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3678   }
3679 }

```

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

`\symref`
`\symname`

```

3680 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3681
3682 \keys_define:nn { stex / symname } {
3683   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3684   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3685   root     .tl_set_x:N      = \l__stex_terms_root_tl
3686 }
3687
3688 \cs_new_protected:Nn \stex_symname_args:n {
3689   \tl_clear:N \l__stex_terms_post_tl
3690   \tl_clear:N \l__stex_terms_pre_tl
3691   \tl_clear:N \l__stex_terms_root_str
3692   \keys_set:nn { stex / symname } { #1 }
3693 }
3694
3695 \NewDocumentCommand \symref { m m }{
3696   \let\compemph_uri_prev:\compemph@uri
3697   \let\compemph@uri\symrefemph@uri
3698   \STEXsymbol{#1}!\{ #2 }
3699   \let\compemph@uri\compemph_uri_prev:
3700 }
3701
3702 \NewDocumentCommand \synonym { O{} m m }{
3703   \stex_symname_args:n { #1 }
3704   \let\compemph_uri_prev:\compemph@uri
3705   \let\compemph@uri\symrefemph@uri
3706   % TODO
3707   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3708   \let\compemph@uri\compemph_uri_prev:

```

```

3709 }
3710
3711 \NewDocumentCommand \symname { 0{} m }{
3712   \stex_symname_args:n { #1 }
3713   \stex_get_symbol:n { #2 }
3714   \str_set:Nx \l_tmpa_str {
3715     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3716   }
3717   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3718
3719   \let\compemph_uri_prev:\compemph@uri
3720   \let\compemph@uri\symrefemph@uri
3721   \exp_args:NNx \use:nn
3722   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3723     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3724   } }
3725   \let\compemph@uri\compemph_uri_prev:
3726 }
3727
3728 \NewDocumentCommand \Symname { 0{} m }{
3729   \stex_symname_args:n { #1 }
3730   \stex_get_symbol:n { #2 }
3731   \str_set:Nx \l_tmpa_str {
3732     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3733   }
3734   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3735   \let\compemph_uri_prev:\compemph@uri
3736   \let\compemph@uri\symrefemph@uri
3737   \exp_args:NNx \use:nn
3738   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3739     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3740     \l__stex_terms_post_tl
3741   } }
3742   \let\compemph@uri\compemph_uri_prev:
3743 }

```

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

30.3 Notation Components

```

3744 <@@=stex_notationcomps>

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

3745 \cs_new_protected:Npn \_comp #1 {
3746   \str_if_empty:NF \l_stex_current_symbol_str {
3747     \stex_html_backend:TF {
3748       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3749     }{
3750       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3751     }
3752   }
3753 }
3754
3755 \cs_new_protected:Npn \_varcomp #1 {

```

```

3756 \str_if_empty:NF \l_stex_current_symbol_str {
3757   \stex_html_backend:TF {
3758     \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3759   }{
3760     \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3761   }
3762 }
3763 }
3764
3765 \def\comp{\_comp}
3766
3767 \cs_new_protected:Npn \compemph@uri #1 #2 {
3768   \compemph{ #1 }
3769 }
3770
3771
3772 \cs_new_protected:Npn \compemph #1 {
3773   #1
3774 }
3775
3776 \cs_new_protected:Npn \defemph@uri #1 #2 {
3777   \defemph{#1}
3778 }
3779
3780 \cs_new_protected:Npn \defemph #1 {
3781   \textbf{#1}
3782 }
3783
3784 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3785   \symrefemph{#1}
3786 }
3787
3788 \cs_new_protected:Npn \symrefemph #1 {
3789   \emph{#1}
3790 }
3791
3792 \cs_new_protected:Npn \varemp@uri #1 #2 {
3793   \varemp{#1}
3794 }
3795
3796 \cs_new_protected:Npn \varemp #1 {
3797   #1
3798 }

```

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

\ellipses

```

3799 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3800 \bool_new:N \l_stex_inparray_bool
3801 \bool_set_false:N \l_stex_inparray_bool
3802 \NewDocumentCommand \parray { m m } {

```

```

3803 \begingroup
3804 \bool_set_true:N \l_stex_inarray_bool
3805 \begin{array}{#1}
3806 #2
3807 \end{array}
3808 \endgroup
3809 }
3810
3811 \NewDocumentCommand \prmatrix { m } {
3812 \begingroup
3813 \bool_set_true:N \l_stex_inarray_bool
3814 \begin{matrix}
3815 #1
3816 \end{matrix}
3817 \endgroup
3818 }
3819
3820 \def \maybepline {
3821 \bool_if:NT \l_stex_inarray_bool {\hline}
3822 }
3823
3824 \def \parrayline #1 #2 {
3825 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3826 }
3827
3828 \def \pmrow #1 { \parrayline{}{ #1 } }
3829
3830 \def \parraylineh #1 #2 {
3831 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3832 }
3833
3834 \def \parraycell #1 {
3835 #1 \bool_if:NT \l_stex_inarray_bool {&}
3836 }

```

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

30.4 Variables

```

3837 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3838 \cs_new_protected:Nn \stex_invoke_variable:n {
3839 \if_mode_math:
3840 \exp_after:wN \__stex_variables_invoke_math:n
3841 \else:
3842 \exp_after:wN \__stex_variables_invoke_text:n
3843 \fi: {#1}
3844 }
3845
3846 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3847 %TODO
3848 }
3849

```

```

3850
3851 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3852   \peek_charcode_remove:NTF ! {
3853     \peek_charcode_remove:NTF ! {
3854       \peek_charcode:NTF [ {
3855         \__stex_variables_invoke_op_custom:nw
3856       }{
3857         % TODO throw error
3858       }
3859     }{
3860       \__stex_variables_invoke_op:n { #1 }
3861     }
3862   }{
3863     \peek_charcode_remove:NTF * {
3864       \__stex_variables_invoke_text:n { #1 }
3865     }{
3866       \__stex_variables_invoke_math_ii:n { #1 }
3867     }
3868   }
3869 }
3870
3871 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3872   \cs_if_exist:cTF {
3873     stex_var_op_notation_ #1 _cs
3874   }{
3875     \exp_args:Nnx \use:nn {
3876       \def\comp{\_varcomp}
3877       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3878       \_stex_term_omv:nn { var://#1 }{
3879         \use:c{stex_var_op_notation_ #1 _cs }
3880       }
3881     }{
3882       \_stex_reset:N \comp
3883       \_stex_reset:N \l_stex_current_symbol_str
3884     }
3885   }{
3886     \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
3887       \__stex_variables_invoke_math_ii:n {#1}
3888     }{
3889       \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
3890     }
3891   }
3892 }
3893
3894 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3895   \cs_if_exist:cTF {
3896     stex_var_notation_#1_cs
3897   }{
3898     \tl_set:Nx \stex_symbol_after_invokation_tl {
3899       \_stex_reset:N \comp
3900       \_stex_reset:N \stex_symbol_after_invokation_tl
3901       \_stex_reset:N \l_stex_current_symbol_str
3902       \bool_set_true:N \l_stex_allow_semantic_bool
3903     }

```



```

3904     \def\comp{\_varcomp}
3905     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3906     \bool_set_false:N \l_stex_allow_semantic_bool
3907     \use:c{stex_var_notation_#1_cs}
3908   }{
3909     \msg_error:nxx{stex}{error/nonotation}{variable-#1}{s}
3910   }
3911 }

```

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

30.5 Sequences

```

3912 <@@=stex_sequences>
3913
3914 \cs_new_protected:Nn \stex_invoke_sequence:n {
3915   \peek_charcode_remove:NTF ! {
3916     \_stex_term_omv:nn {varseq://#1}{
3917       \exp_args:Nnx \use:nn {
3918         \def\comp{\_varcomp}
3919         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3920         \prop_item:cn{stex_varseq_#1_prop}{notation}
3921       }{
3922         \_stex_reset:N \comp
3923         \_stex_reset:N \l_stex_current_symbol_str
3924       }
3925     }
3926   }{
3927     \bool_set_false:N \l_stex_allow_semantic_bool
3928     \def\comp{\_varcomp}
3929     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3930     \tl_set:Nx \stex_symbol_after_invokation_tl {
3931       \_stex_reset:N \comp
3932       \_stex_reset:N \stex_symbol_after_invokation_tl
3933       \_stex_reset:N \l_stex_current_symbol_str
3934       \bool_set_true:N \l_stex_allow_semantic_bool
3935     }
3936     \use:c { stex_varseq_#1_cs }
3937   }
3938 }
3939 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3940 ⟨*package⟩
3941
3942 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3943
3944     Warnings and error messages
3945 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3946     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3947 }
3948 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3949     Symbol~#1~not~assigned~in~interpretmodule~#2
3950 }
3951 \msg_new:nnn{stex}{error/unknownstructure}{
3952     No~structure~#1~found!
3953 }
3954 \msg_new:nnn{stex}{error/unknownfield}{
3955     No~field~#1~in~instance~#2~found!~\#3
3956 }
3957
3958 \msg_new:nnn{stex}{error/keyval}{
3959     Invalid~key=value~pair~#1
3960 }
3961 \msg_new:nnn{stex}{error/instantiate/missing}{
3962     Assignments~missing~in~instantiate:~#1
3963 }
3964 \msg_new:nnn{stex}{error/incompatible}{
3965     Incompatible~signature:~#1~(#2)~and~#3~(#4)
3966 }
3967
3968
```

31.1 Imports with modification

```

3969 <@@=stex_copymodule>
3970 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3971   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3972     \tl_set:Nn \l_tmpa_tl { #1 }
3973     \__stex_copymodule_get_symbol_from_cs:
3974   }{
3975     % argument is a string
3976     % is it a command name?
3977     \cs_if_exist:cTF { #1 }{
3978       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3979       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3980       \str_if_empty:NNTF \l_tmpa_str {
3981         \exp_args:Nx \cs_if_eq:NNTF {
3982           \tl_head:N \l_tmpa_tl
3983         } \stex_invoke_symbol:n {
3984           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3985         }{
3986           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3987         }
3988       } {
3989         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3990       }
3991     }{
3992       % argument is not a command name
3993       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3994       % \l_stex_all_symbols_seq
3995     }
3996   }
3997 }
3998
3999 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4000   \str_set:Nn \l_tmpa_str { #1 }
4001   \bool_set_false:N \l_tmpa_bool
4002   \bool_if:NF \l_tmpa_bool {
4003     \tl_set:Nn \l_tmpa_tl {
4004       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4005     }
4006     \str_set:Nn \l_tmpa_str { #1 }
4007     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4008     \seq_map_inline:Nn #2 {
4009       \str_set:Nn \l_tmpb_str { ##1 }
4010       \str_if_eq:eeT { \l_tmpa_str } {
4011         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4012       } {
4013         \seq_map_break:n {
4014           \tl_set:Nn \l_tmpa_tl {
4015             \str_set:Nn \l_stex_get_symbol_uri_str {
4016               ##1
4017             }
4018           }
4019         }
4020       }

```

```

4021     }
4022     \l_tmpa_tl
4023   }
4024 }
4025
4026 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4027   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4028     { \tl_tail:N \l_tmpa_tl }
4029   \tl_if_single:NTF \l_tmpa_tl {
4030     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4031       \exp_after:wN \str_set:Nn \exp_after:wN
4032         \l_stex_get_symbol_uri_str \l_tmpa_tl
4033       \__stex_copymodule_get_symbol_check:n { #1 }
4034     }{
4035       % TODO
4036       % tail is not a single group
4037     }
4038   }{
4039     % TODO
4040     % tail is not a single group
4041   }
4042 }
4043
4044 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4045   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4046     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4047       :~\seq_use:Nn #1 {,~}
4048     }
4049   }
4050 }
4051
4052 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4053   % import module
4054   \stex_import_module_uri:nn { #1 } { #2 }
4055   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4056   \stex_import_require_module:nnnn
4057     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4058     { \l_stex_import_path_str } { \l_stex_import_name_str }
4059
4060   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4061   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4062
4063   % fields
4064   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4065   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4066     \seq_map_inline:cn {c_stex_module_##1_constants}{
4067       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4068         ##1 ? #####1
4069       }
4070     }
4071   }
4072
4073   % setup prop
4074   \seq_clear:N \l_tmpa_seq

```

```

4075 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4076   name      = \l_stex_current_copymodule_name_str ,
4077   module    = \l_stex_current_module_str ,
4078   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4079   includes  = \l_tmpa_seq %,
4080 % fields    = \l_tmpa_seq
4081 }
4082 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4083   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4084 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4085 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4086
4087 \stex_if_do_html:T {
4088   \begin{stex_annotate_env} {#4} {
4089     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4090   }
4091   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
4092 }
4093 }
4094
4095 \cs_new_protected:Nn \stex_copymodule_end:n {
4096   % apply to every field
4097   \def \l_tmpa_cs ##1 ##2 {#1}
4098
4099   \tl_clear:N \__stex_copymodule_module_tl
4100   \tl_clear:N \__stex_copymodule_exec_tl
4101
4102   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4103   \seq_clear:N \__stex_copymodule_fields_seq
4104
4105   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4106     \seq_map_inline:cn {c_stex_module_##1_constants}{
4107
4108       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4109       \l_tmpa_cs{##1}{####1}
4110
4111       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4112         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4113         \stex_if_do_html:T {
4114           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4115             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4116           }
4117         }
4118       }{
4119         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4120       }
4121
4122       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1_prop}
4123       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4124       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4125
4126       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4127         \stex_if_do_html:T {
4128           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4129         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4130     }
4131 }
4132 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4133 }
4134
4135 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4136 \tl_put_right:Nx \__stex_copymodule_module_tl {
4137     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4138     \prop_set_from_keyval:cn {
4139         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4140     }{
4141         \prop_to_keyval:N \l_tmpa_prop
4142     }
4143 }
4144
4145 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4146     \stex_if_do_html:T {
4147         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4148             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4149         }
4150     }
4151     \tl_put_right:Nx \__stex_copymodule_module_tl {
4152         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4153             \stex_invoke_symbol:n {
4154                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4155             }
4156         }
4157     }
4158 }
4159
4160 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4161
4162 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4163     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4164 }
4165
4166 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4167     \stex_if_do_html:TF{
4168         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4169     }{
4170         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4171     }
4172 }
4173 }
4174 }
4175
4176
4177 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4178 \tl_put_left:Nx \__stex_copymodule_module_tl {
4179     \prop_set_from_keyval:cn {
4180         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4181     }{
4182         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4183     }
4184   }
4185
4186   \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4187     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4188   }
4189
4190   \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4191   \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4192   \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4193
4194   \__stex_copymodule_exec_tl
4195   \stex_if_do_html:T {
4196     \end{stex_annotate_env}
4197   }
4198 }
4199
4200 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4201   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4202   \stex_deactivate_macro:Nn \symdecl {module~environments}
4203   \stex_deactivate_macro:Nn \symdef {module~environments}
4204   \stex_deactivate_macro:Nn \notation {module~environments}
4205   \stex_reactivate_macro:N \assign
4206   \stex_reactivate_macro:N \renamedekl
4207   \stex_reactivate_macro:N \donotcopy
4208   \stex_smsmode_do:
4209 }{
4210   \stex_copymodule_end:n {}
4211 }
4212
4213 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4214   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4215   \stex_deactivate_macro:Nn \symdecl {module~environments}
4216   \stex_deactivate_macro:Nn \symdef {module~environments}
4217   \stex_deactivate_macro:Nn \notation {module~environments}
4218   \stex_reactivate_macro:N \assign
4219   \stex_reactivate_macro:N \renamedekl
4220   \stex_reactivate_macro:N \donotcopy
4221   \stex_smsmode_do:
4222 }{
4223   \stex_copymodule_end:n {
4224     \tl_if_exist:cF {
4225       l__stex_copymodule_copymodule_##1?##2_def_tl
4226     }{
4227       \str_if_eq:eeF {
4228         \prop_item:cn{
4229           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4230         }{ true }{
4231           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4232             ##1?##2
4233           }{\l_stex_current_copymodule_name_str}
4234         }
4235       }
4236     }

```

```

4237 }
4238
4239 \iffalse \begin{stex_annotate_env} \fi
4240 \NewDocumentEnvironment {realization} { 0 } { m } {
4241   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4242   \stex_deactivate_macro:Nn \symdecl {module~environments}
4243   \stex_deactivate_macro:Nn \symdef {module~environments}
4244   \stex_deactivate_macro:Nn \notation {module~environments}
4245   \stex_reactivate_macro:N \donotcopy
4246   \stex_reactivate_macro:N \assign
4247   \stex_smsmode_do:
4248 } {
4249   \stex_import_module_uri:nn { #1 } { #2 }
4250   \tl_clear:N \__stex_copymodule_exec_tl
4251   \tl_set:Nx \__stex_copymodule_module_tl {
4252     \stex_import_require_module:nnnn
4253     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4254     { \l_stex_import_path_str } { \l_stex_import_name_str }
4255   }
4256
4257   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4258     \seq_map_inline:cn {c_stex_module_##1_constants}{
4259       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4260       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4261         \stex_if_do_html:T {
4262           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4263             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4264               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4265             }
4266           }
4267         }
4268         \tl_put_right:Nx \__stex_copymodule_module_tl {
4269           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4270         }
4271       }
4272     }
4273
4274     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4275
4276     \__stex_copymodule_exec_tl
4277     \stex_if_do_html:T {\end{stex_annotate_env}}
4278   }
4279
4280 \NewDocumentCommand \donotcopy { m } {
4281   \str_clear:N \l_stex_import_name_str
4282   \str_set:Nn \l_tmpa_str { #1 }
4283   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4284   \seq_map_inline:Nn \l_stex_all_modules_seq {
4285     \str_set:Nn \l_tmpb_str { ##1 }
4286     \str_if_eq:eeT { \l_tmpa_str } {
4287       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4288     } {
4289       \seq_map_break:n {
4290         \stex_if_do_html:T {

```



```

4291         \stex_if_smsmode:F {
4292             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4293                 \stex_annotate:nnn{domain}{##1}{}}
4294         }
4295     }
4296 }
4297 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4298 }
4299 }
4300 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4301     \str_set:Nn \l_tmpb_str { #####1 }
4302     \str_if_eq:eeT { \l_tmpa_str } {
4303         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4304     } {
4305         \seq_map_break:n {\seq_map_break:n {
4306             \stex_if_do_html:T {
4307                 \stex_if_smsmode:F {
4308                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4309                         \stex_annotate:nnn{domain}{
4310                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4311                         }{}
4312                     }
4313                 }
4314             }
4315             \str_set:Nx \l_stex_import_name_str {
4316                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4317             }
4318         }}
4319     }
4320 }
4321 }
4322 \str_if_empty:NTF \l_stex_import_name_str {
4323     % TODO throw error
4324 }{
4325     \stex_collect_imports:n {\l_stex_import_name_str }
4326     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4327         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4328         \seq_map_inline:cn {c_stex_module_###1_constants}{
4329             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4330             \bool_lazy_any:nT {
4331                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_name_str}}
4332                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_macroname_str}}
4333                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_def_tl}}
4334             }{
4335                 % TODO throw error
4336             }
4337         }
4338     }
4339     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4340     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4341     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4342 }
4343 \stex_smsmode_do:
4344 }

```

```

4345
4346 \NewDocumentCommand \assign { m m }{
4347   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4348   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4349   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4350   \stex_smsmode_do:
4351 }
4352
4353 \keys_define:nn { stex / renamedecl } {
4354   name          .str_set_x:N = \l_stex_renamedecl_name_str
4355 }
4356 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4357   \str_clear:N \l_stex_renamedecl_name_str
4358   \keys_set:nn { stex / renamedecl } { #1 }
4359 }
4360
4361 \NewDocumentCommand \renamedecl { O{} m m }{
4362   \__stex_copymodule_renamedecl_args:n { #1 }
4363   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4364   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4365   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4366   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4367     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4368       \l_stex_get_symbol_uri_str
4369     } }
4370   } {
4371     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4372       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4373       \prop_set_eq:cc {l_stex_symdecl_
4374         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4375       _prop
4376       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4377       \seq_set_eq:cc {l_stex_symdecl_
4378         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4379       _notations
4380       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4381       \prop_put:cnx {l_stex_symdecl_
4382         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4383       _prop
4384       }{ name }{ \l_stex_renamedecl_name_str }
4385       \prop_put:cnx {l_stex_symdecl_
4386         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4387       _prop
4388       }{ module }{ \l_stex_current_module_str }
4389       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4390         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4391       }
4392       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4393         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4394       } }
4395     }
4396   \stex_smsmode_do:
4397 }
4398

```

```

4399 \stex_deactivate_macro:Nn \assign {copymodules}
4400 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4401 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4402
4403

```

31.2 The feature environment

structural@feature

```

4404 <@@=stex_features>
4405
4406 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4407   \stex_if_in_module:F {
4408     \msg_set:nnn{stex}{error/nomodule}{
4409       Structural~Feature~has~to~occur~in~a~module:\\
4410       Feature~#2~of~type~#1\\
4411       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4412     }
4413     \msg_error:nn{stex}{error/nomodule}
4414   }
4415
4416   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4417
4418   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4419
4420   \stex_if_do_html:T {
4421     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4422     \stex_annotate_invisible:nnn{header}{\{ #3 }
4423   }
4424   }{
4425     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4426     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4427     \stex_debug:nn{features}{
4428       Feature: \l_stex_last_feature_str
4429     }
4430     \stex_if_do_html:T {
4431       \end{stex_annotate_env}
4432     }
4433   }

```

31.3 Structure

structure

```

4434 <@@=stex_structures>
4435 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4436   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4437     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4438   }
4439   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4440   {#1}{#2}
4441 }
4442

```

```

4443 \keys_define:nn { stex / features / structure } {
4444   name          .str_set_x:N = \l__stex_structures_name_str ,
4445 }
4446
4447 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4448   \str_clear:N \l__stex_structures_name_str
4449   \keys_set:nn { stex / features / structure } { #1 }
4450 }
4451
4452 \NewDocumentEnvironment{mathstructure}{m O{}}{
4453   \__stex_structures_structure_args:n { #2 }
4454   \str_if_empty:NT \l__stex_structures_name_str {
4455     \str_set:Nx \l__stex_structures_name_str { #1 }
4456   }
4457   \stex_suppress_html:n {
4458     \exp_args:Nx \stex_symdecl_do:nn {
4459       name = \l__stex_structures_name_str ,
4460       def  = {\STEXsymbol{module-type}}{
4461         \stex_term_math_oms:nnnn {
4462           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4463             { ns } ?
4464           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4465             { name } / \l__stex_structures_name_str - structure
4466         }{}{0}{}
4467       }}
4468     }{ #1 }
4469   }
4470   \exp_args:Nnnx
4471   \begin{structural_feature_module}{ structure }
4472     { \l__stex_structures_name_str }{}
4473   \stex_smsmode_do:
4474 }{
4475   \end{structural_feature_module}
4476   \stex_reset_up_to_module:n \l_stex_last_feature_str
4477   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4478   \seq_clear:N \l_tmpa_seq
4479   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4480     \seq_map_inline:cn{c_stex_module_##1_constants}{
4481       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4482     }
4483   }
4484   \exp_args:Nnno
4485   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4486   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4487   \stex_add_structure_to_current_module:nn
4488     \l__stex_structures_name_str
4489     \l_stex_last_feature_str
4490
4491   \stex_execute_in_module:x {
4492     \tl_set:cn { #1 }{
4493       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4494     }
4495   }
4496 }

```

```

4497
4498 \cs_new:Nn \stex_invoke_structure:nn {
4499   \stex_invoke_symbol:n { #1?#2 }
4500 }
4501
4502 \cs_new_protected:Nn \stex_get_structure:n {
4503   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4504     \tl_set:Nn \l_tmpa_tl { #1 }
4505     \__stex_structures_get_from_cs:
4506   }{
4507     \cs_if_exist:cTF { #1 }{
4508       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4509       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4510       \str_if_empty:NNTF \l_tmpa_str {
4511         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4512           \__stex_structures_get_from_cs:
4513         }{
4514           \__stex_structures_get_from_string:n { #1 }
4515         }
4516       }{
4517         \__stex_structures_get_from_string:n { #1 }
4518       }
4519     }{
4520       \__stex_structures_get_from_string:n { #1 }
4521     }
4522   }
4523 }
4524
4525 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4526   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4527     { \tl_tail:N \l_tmpa_tl }
4528   \str_set:Nx \l_tmpa_str {
4529     \exp_after:wN \use_i:nn \l_tmpa_tl
4530   }
4531   \str_set:Nx \l_tmpb_str {
4532     \exp_after:wN \use_ii:nn \l_tmpa_tl
4533   }
4534   \str_set:Nx \l_stex_get_structure_str {
4535     \l_tmpa_str ? \l_tmpb_str
4536   }
4537   \str_set:Nx \l_stex_get_structure_module_str {
4538     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4539   }
4540 }
4541
4542 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4543   \tl_set:Nn \l_tmpa_tl {
4544     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4545   }
4546   \str_set:Nn \l_tmpa_str { #1 }
4547   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4548
4549   \seq_map_inline:Nn \l_stex_all_modules_seq {
4550     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4551 \prop_map_inline:cn {c_stex_module_##1_structures} {
4552   \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4553     \prop_map_break:n{\seq_map_break:n{
4554       \tl_set:Nn \l_tmpa_tl {
4555         \str_set:Nn \l_stex_get_structure_str {##1?####1}
4556         \str_set:Nn \l_stex_get_structure_module_str {####2}
4557       }
4558     }}
4559   }
4560 }
4561 }
4562 }
4563 \l_tmpa_tl
4564 }

```

\instantiate

```

4565
4566 \keys_define:nn { stex / instantiate } {
4567   name          .str_set_x:N = \l__stex_structures_name_str
4568 }
4569 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4570   \str_clear:N \l__stex_structures_name_str
4571   \keys_set:nn { stex / instantiate } { #1 }
4572 }
4573
4574 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4575   \begin{group}
4576     \stex_get_structure:n {#3}
4577     \__stex_structures_instantiate_args:n { #2 }
4578     \str_if_empty:NT \l__stex_structures_name_str {
4579       \str_set:Nn \l__stex_structures_name_str { #1 }
4580     }
4581     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4582     \seq_clear:N \l__stex_structures_fields_seq
4583     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4584     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4585       \seq_map_inline:cn {c_stex_module_##1_constants}{
4586         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4587       }
4588     }
4589
4590     \tl_if_empty:nF{#5}{
4591       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4592       \prop_clear:N \l_tmpa_prop
4593       \seq_map_inline:Nn \l_tmpa_seq {
4594         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4595         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4596           \msg_error:nnn{stex}{error/keyval}{##1}
4597         }
4598         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4599         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4600         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4601         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4602         \exp_args:Nxx \str_if_eq:nnF

```

```

4603         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4604         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4605         \msg_error:nnxxxx{stex}{error/incompatible}
4606         {\l__stex_structures_dom_str}
4607         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4608         {\l_stex_get_symbol_uri_str}
4609         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4610     }
4611     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4612 }
4613 }
4614
4615 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4616     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4617     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4618
4619     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4620     \stex_execute_in_module:x {
4621         \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
4622         name = \l_tmpa_str ,
4623         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4624         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4625         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4626     }
4627     \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4628 }
4629
4630 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4631     \stex_find_notation:nn{##1}{}
4632     \stex_execute_in_module:x {
4633         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4634     }
4635
4636     \stex_copy_control_sequence_ii:ccN
4637     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4638     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4639     \l_tmpa_tl
4640     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4641
4642
4643     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4644         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4645         \stex_execute_in_module:x {
4646             \tl_set:cn
4647             {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4648             { \exp_args:No \exp_not:n \l_tmpa_cs}
4649         }
4650     }
4651
4652 }
4653
4654 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4655 }
4656

```

```

4657 \stex_execute_in_module:x {
4658   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4659   domain = \l_stex_get_structure_module_str ,
4660   \prop_to_keyval:N \l_tmpa_prop
4661 }
4662 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4663 }
4664 \stex_debug:nn{instantiate}{
4665   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4666   \prop_to_keyval:N \l_tmpa_prop
4667 }
4668 \exp_args:Nxx \stex_symdecl_do:nn {
4669   type={\STEXsymbol{module-type}}{
4670     \stex_term_math_oms:nnnn {
4671       \l_stex_get_structure_module_str
4672     }{}{0}{}
4673   }}
4674 }{\l__stex_structures_name_str}
4675 % {
4676   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4677   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4678   \stex_notation_do:nnnnn{}{}{0}{}{\comp{#4}}
4679 % }
4680 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4681 \endgroup
4682 \stex_smsmode_do:\ignorespacesandpars
4683 }
4684
4685 \cs_new_protected:Nn \stex_symbol_or_var:n {
4686   \cs_if_exist:cTF{#1}{
4687     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4688     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4689     \str_if_empty:NTF \l_tmpa_str {
4690       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4691       \stex_invoke_variable:n {
4692         \bool_set_true:N \l_stex_symbol_or_var_bool
4693         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4694         \str_set:Nx \l_stex_get_symbol_uri_str {
4695           \exp_after:wN \use:n \l_tmpa_tl
4696         }
4697       }{
4698         \bool_set_false:N \l_stex_symbol_or_var_bool
4699         \stex_get_symbol:n{#1}
4700       }
4701     }{
4702       \__stex_structures_symbolorvar_from_string:n{ #1 }
4703     }
4704   }{
4705     \__stex_structures_symbolorvar_from_string:n{ #1 }
4706   }
4707 }
4708
4709 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4710   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```



```

4711     \bool_set_true:N \l_stex_symbol_or_var_bool
4712     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4713   }{
4714     \bool_set_false:N \l_stex_symbol_or_var_bool
4715     \stex_get_symbol:n{#1}
4716   }
4717 }
4718
4719 \keys_define:nn { stex / varinstantiate } {
4720   name      .str_set_x:N = \l__stex_structures_name_str,
4721   bind      .choices:nn =
4722     {forall,exists}
4723     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4724 }
4725
4726 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4727   \str_clear:N \l__stex_structures_name_str
4728   \str_clear:N \l__stex_structures_bind_str
4729   \keys_set:nn { stex / varinstantiate } { #1 }
4730 }
4731
4732 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4733   \begingroup
4734     \stex_get_structure:n {#3}
4735     \__stex_structures_varinstantiate_args:n { #2 }
4736     \str_if_empty:NT \l__stex_structures_name_str {
4737       \str_set:Nn \l__stex_structures_name_str { #1 }
4738     }
4739     \stex_if_do_html:TF{
4740       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4741     }{\use:n}
4742     {
4743       \stex_if_do_html:T{
4744         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4745       }
4746       \seq_clear:N \l__stex_structures_fields_seq
4747       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4748       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4749         \seq_map_inline:cn {c_stex_module_##1_constants}{
4750           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4751         }
4752       }
4753       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4754       \prop_clear:N \l_tmpa_prop
4755       \tl_if_empty:nF {#5} {
4756         \seq_set_split:Nnn \l_tmpa_seq , {#5}
4757         \seq_map_inline:Nn \l_tmpa_seq {
4758           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4759           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4760             \msg_error:nnn{stex}{error/keyval}{##1}
4761           }
4762           \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4763           \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4764           \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```

```

4765 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4766 \stex_if_do_html:T{
4767   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4768 }
4769 \bool_if:NTF \l_stex_symbol_or_var_bool {
4770   \exp_args:Nxx \str_if_eq:nnF
4771     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4772     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4773     \msg_error:nnxxxx{stex}{error/incompatible}
4774     {\l__stex_structures_dom_str
4775     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4776     {\l_stex_get_symbol_uri_str
4777     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4778   }
4779   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4780 }{
4781   \exp_args:Nxx \str_if_eq:nnF
4782     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4783     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4784     \msg_error:nnxxxx{stex}{error/incompatible}
4785     {\l__stex_structures_dom_str
4786     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4787     {\l_stex_get_symbol_uri_str
4788     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4789   }
4790   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4791 }
4792 }
4793 }
4794 \tl_gclear:N \g__stex_structures_aftergroup_tl
4795 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4796   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_fields_seq #1} {}}
4797   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4798   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4799     \stex_find_notation:nn{##1}{}
4800     \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4801       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4802     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4803     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4804       \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4805       {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4806       \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4807     }
4808   }
4809 }
4810 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4811   \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4812     name = \l_tmpa_str ,
4813     args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4814     arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4815     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4816   }
4817   \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4818   {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4819         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4820         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4821     }
4822     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4823 }
4824     \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4825         \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4826             domain = \l_stex_get_structure_module_str ,
4827             \prop_to_keyval:N \l_tmpa_prop
4828         }
4829         \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4830         \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4831             \exp_args:Nnx \exp_not:N \use:nn {
4832                 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4833                 \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4834                     \exp_not:n{
4835                         \_varcomp{#4}
4836                     }
4837                 }
4838             }{
4839                 \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4840             }
4841         }
4842     }
4843 }
4844 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4845 \aftergroup\g__stex_structures_aftergroup_tl
4846 \endgroup
4847 \stex_smsmode_do:\ignorespacesandpars
4848 }
4849
4850 \cs_new_protected:Nn \stex_invoke_instance:n {
4851     \peek_charcode_remove:NTF ! {
4852         \stex_invoke_symbol:n{#1}
4853     }{
4854         \_stex_invoke_instance:nn {#1}
4855     }
4856 }
4857
4858
4859 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4860     \peek_charcode_remove:NTF ! {
4861         \exp_args:Nnx \use:nn {
4862             \def\comp{\_varcomp}
4863             \use:c{l_stex_varinstance_#1_op_tl}
4864         }{
4865             \_stex_reset:N \comp
4866         }
4867     }{
4868         \_stex_invoke_varinstance:nn {#1}
4869     }
4870 }
4871
4872 \cs_new_protected:Nn \stex_invoke_instance:nn {

```

```

4873 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4874   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4875 }{
4876   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4877   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4878     \prop_to_keyval:N \l_tmpa_prop
4879   }
4880 }
4881 }
4882
4883 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4884   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4885     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4886     \l_tmpa_tl
4887   }{
4888     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4889   }
4890 }

```

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

`\stex_invoke_structure:nnn`

```

4891 % #1: URI of the instance
4892 % #2: URI of the instantiated module
4893 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4894   \tl_if_empty:nTF{ #3 }{
4895     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4896       c_stex_feature_ #2 _prop
4897     }
4898     \tl_clear:N \l_tmpa_tl
4899     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4900     \seq_map_inline:Nn \l_tmpa_seq {
4901       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4902       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4903       \cs_if_exist:cT {
4904         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4905       }{
4906         \tl_if_empty:NF \l_tmpa_tl {
4907           \tl_put_right:Nn \l_tmpa_tl {,}
4908         }
4909         \tl_put_right:Nx \l_tmpa_tl {
4910           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4911         }
4912       }
4913     }
4914     \exp_args:No \mathstruct \l_tmpa_tl
4915   }{
4916     \stex_invoke_symbol:n{#1/#3}
4917   }
4918 }

```

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

```

4919 </package>

```

Chapter 32

STEX -Statements Implementation

```
4920 <*package>
4921
4922 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4923
4924 <@@=stex_statements>
    Warnings and error messages
4925

\titleemph
4926 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
4927 \keys_define:nn {stex / definiendum }{
4928   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4929   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4930   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4931   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4932 }
4933 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4934   \str_clear:N \l__stex_statements_definiendum_root_str
4935   \tl_clear:N \l__stex_statements_definiendum_post_tl
4936   \str_clear:N \l__stex_statements_definiendum_gfa_str
4937   \keys_set:nn { stex / definiendum }{ #1 }
4938 }
4939 \NewDocumentCommand \definiendum { O{} m m } {
4940   \__stex_statements_definiendum_args:n { #1 }
4941   \stex_get_symbol:n { #2 }
4942   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4943   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4944     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4945     \tl_set:Nn \l_tmpa_tl { #3 }
4946   } {
4947     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4948     \tl_set:Nn \l_tmpa_tl {
4949       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4950     }
4951   }
4952 } {
4953   \tl_set:Nn \l_tmpa_tl { #3 }
4954 }
4955
4956 % TODO root
4957 \stex_html_backend:TF {
4958   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4959 } {
4960   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4961 }
4962 }
4963 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4964
4965 \NewDocumentCommand \definame { 0{ } m } {
4966   \__stex_statements_definiendum_args:n { #1 }
4967   % TODO: root
4968   \stex_get_symbol:n { #2 }
4969   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4970   \str_set:Nx \l_tmpa_str {
4971     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4972   }
4973   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4974   \stex_html_backend:TF {
4975     \stex_if_do_html:T {
4976       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4977         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4978       }
4979     }
4980   } {
4981     \exp_args:Nnx \defemph@uri {
4982       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4983     } { \l_stex_get_symbol_uri_str }
4984   }
4985 }
4986 \stex_deactivate_macro:Nn \definame {definition~environments}
4987
4988 \NewDocumentCommand \Definame { 0{ } m } {
4989   \__stex_statements_definiendum_args:n { #1 }
4990   \stex_get_symbol:n { #2 }
4991   \str_set:Nx \l_tmpa_str {
4992     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4993   }
4994   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

4995 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4996 \stex_html_backend:TF {
4997   \stex_if_do_html:T {
4998     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4999       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5000     }
5001   }
5002 } {
5003   \exp_args:Nnx \defemph@uri {
5004     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5005   } { \l_stex_get_symbol_uri_str }
5006 }
5007 }
5008 \stex_deactivate_macro:Nn \Definame {definition~environments}
5009
5010 \NewDocumentCommand \premise { m }{
5011   \stex_annotate:nnn{ premise }{}{ #1 }
5012 }
5013 \NewDocumentCommand \conclusion { m }{
5014   \stex_annotate:nnn{ conclusion }{}{ #1 }
5015 }
5016 \NewDocumentCommand \definiens { 0{} m }{
5017   \str_clear:N \l_stex_get_symbol_uri_str
5018   \tl_if_empty:nF {#1} {
5019     \stex_get_symbol:n { #1 }
5020   }
5021   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5022     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5023       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5024     }{
5025       % TODO throw error
5026     }
5027   }
5028   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5029   {\l_stex_current_module_str}{
5030     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5031   }{true}{
5032     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5033     \exp_args:Nx \stex_add_to_current_module:n {
5034       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5035     }
5036   }
5037 }
5038 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5039 }
5040
5041 \stex_deactivate_macro:Nn \premise {definition~example~or~assertion~environments}
5042 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5043 \stex_deactivate_macro:Nn \definiens {definition~environments}
5044

```

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

sdefinition

```

5045
5046 \keys_define:nn {stex / sdefinition }{
5047   type      .str_set_x:N = \sdefinitiontype,
5048   id        .str_set_x:N = \sdefinitionid,
5049   name      .str_set_x:N = \sdefinitionname,
5050   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5051   title     .tl_set:N     = \sdefinitiontitle
5052 }
5053 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5054   \str_clear:N \sdefinitiontype
5055   \str_clear:N \sdefinitionid
5056   \str_clear:N \sdefinitionname
5057   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5058   \tl_clear:N \sdefinitiontitle
5059   \keys_set:nn { stex / sdefinition }{ #1 }
5060 }
5061
5062 \NewDocumentEnvironment{sdefinition}{0{}}{
5063   \__stex_statements_sdefinition_args:n{ #1 }
5064   \stex_reactivate_macro:N \definiendum
5065   \stex_reactivate_macro:N \definame
5066   \stex_reactivate_macro:N \Definame
5067   \stex_reactivate_macro:N \premise
5068   \stex_reactivate_macro:N \definiens
5069   \stex_if_smsmode:F{
5070     \seq_clear:N \l_tmpa_seq
5071     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5072       \tl_if_empty:NF{ ##1 }{
5073         \stex_get_symbol:n { ##1 }
5074         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5075           \l_stex_get_symbol_uri_str
5076         }
5077       }
5078     }
5079     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5080     \exp_args:Nnnx
5081     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {},}}
5082     \str_if_empty:NF \sdefinitiontype {
5083       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5084     }
5085     \str_if_empty:NF \sdefinitionname {
5086       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5087     }
5088     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5089     \tl_clear:N \l_tmpa_tl
5090     \clist_map_inline:Nn \l_tmpa_clist {
5091       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5092         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5093       }
5094     }
5095     \tl_if_empty:NTF \l_tmpa_tl {
5096       \__stex_statements_sdefinition_start:
5097     }{
5098       \l_tmpa_tl

```



```

5099     }
5100   }
5101   \stex_ref_new_doc_target:n \sdefinitionid
5102   \stex_smsmode_do:
5103 }{
5104   \stex_suppress_html:n {
5105     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5106   }
5107   \stex_if_smsmode:F {
5108     \clist_set:No \l_tmpa_clist \sdefinitiontype
5109     \tl_clear:N \l_tmpa_tl
5110     \clist_map_inline:Nn \l_tmpa_clist {
5111       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5112         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5113       }
5114     }
5115     \tl_if_empty:NTF \l_tmpa_tl {
5116       \__stex_statements_sdefinition_end:
5117     }{
5118       \l_tmpa_tl
5119     }
5120     \end{stex_annotate_env}
5121   }
5122 }

```

\stexpatchdefinition

```

5123 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5124   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5125     ~(\sdefinitiontitle)
5126   }~}
5127 }
5128 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5129
5130 \newcommand\stexpatchdefinition[3] [] {
5131   \str_set:Nx \l_tmpa_str{ #1 }
5132   \str_if_empty:NTF \l_tmpa_str {
5133     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5134     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5135   }{
5136     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5137     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5138   }
5139 }

```

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

\inlinedef inline:

```

5140 \keys_define:nn {stex / inlinedef }{
5141   type      .str_set_x:N = \sdefinitiontype,
5142   id        .str_set_x:N = \sdefinitionid,
5143   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5144   name      .str_set_x:N = \sdefinitionname
5145 }
5146 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```

```

5147 \str_clear:N \sdefinitiontype
5148 \str_clear:N \sdefinitionid
5149 \str_clear:N \sdefinitionname
5150 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5151 \keys_set:nn { stex / inlinedef }{ #1 }
5152 }
5153 \NewDocumentCommand \inlinedef { 0{} m } {
5154 \begingroup
5155 \__stex_statements_inlinedef_args:n{ #1 }
5156 \stex_reactivate_macro:N \definiendum
5157 \stex_reactivate_macro:N \definame
5158 \stex_reactivate_macro:N \Definame
5159 \stex_reactivate_macro:N \premise
5160 \stex_reactivate_macro:N \definiens
5161 \stex_ref_new_doc_target:n \sdefinitionid
5162 \stex_if_smsmode:TF{\stex_suppress_html:n {
5163 \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5164 }}{
5165 \seq_clear:N \l_tmpa_seq
5166 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5167 \tl_if_empty:nF{ ##1 }{
5168 \stex_get_symbol:n { ##1 }
5169 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5170 \l_stex_get_symbol_uri_str
5171 }
5172 }
5173 }
5174 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5175 \exp_args:Nnx
5176 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5177 \str_if_empty:NF \sdefinitiontype {
5178 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5179 }
5180 #2
5181 \str_if_empty:NF \sdefinitionname {
5182 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5183 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5184 }
5185 }
5186 }
5187 \endgroup
5188 \stex_smsmode_do:
5189 }

```

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

32.2 Assertions

sassertion

```

5190
5191 \keys_define:nn {stex / sassertion }{
5192 type .str_set_x:N = \sassertiontype,
5193 id .str_set_x:N = \sassertionid,

```

```

5194 title .tl_set:N = \sassertiontitle ,
5195 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5196 name .str_set_x:N = \sassertionname
5197 }
5198 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5199 \str_clear:N \sassertiontype
5200 \str_clear:N \sassertionid
5201 \str_clear:N \sassertionname
5202 \clist_clear:N \l__stex_statements_sassertion_for_clist
5203 \tl_clear:N \sassertiontitle
5204 \keys_set:nn { stex / sassertion }{ #1 }
5205 }
5206
5207 %\tl_new:N \g__stex_statements_aftergroup_tl
5208
5209 \NewDocumentEnvironment{sassertion}{0{}}{
5210 \__stex_statements_sassertion_args:n{ #1 }
5211 \stex_reactivate_macro:N \premise
5212 \stex_reactivate_macro:N \conclusion
5213 \stex_if_smsmode:F {
5214 \seq_clear:N \l_tmpa_seq
5215 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5216 \tl_if_empty:nF{ ##1 }{
5217 \stex_get_symbol:n { ##1 }
5218 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5219 \l_stex_get_symbol_uri_str
5220 }
5221 }
5222 }
5223 \exp_args:Nnnx
5224 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5225 \str_if_empty:NF \sassertiontype {
5226 \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5227 }
5228 \str_if_empty:NF \sassertionname {
5229 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5230 }
5231 \clist_set:Nn \l_tmpa_clist \sassertiontype
5232 \tl_clear:N \l_tmpa_tl
5233 \clist_map_inline:Nn \l_tmpa_clist {
5234 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5235 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5236 }
5237 }
5238 \tl_if_empty:NTF \l_tmpa_tl {
5239 \__stex_statements_sassertion_start:
5240 }{
5241 \l_tmpa_tl
5242 }
5243 }
5244 \str_if_empty:NTF \sassertionid {
5245 \str_if_empty:NF \sassertionname {
5246 \stex_ref_new_doc_target:n { }
5247 }

```

```

5248 } {
5249   \stex_ref_new_doc_target:n \sassertionid
5250 }
5251 \stex_smsmode_do:
5252 ){
5253   \str_if_empty:NF \sassertionname {
5254     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5255     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5256   }
5257   \stex_if_smsmode:F {
5258     \clist_set:Nn \l_tmpa_clist \sassertiontype
5259     \tl_clear:N \l_tmpa_tl
5260     \clist_map_inline:Nn \l_tmpa_clist {
5261       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5262         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5263       }
5264     }
5265     \tl_if_empty:NTF \l_tmpa_tl {
5266       __stex_statements_sassertion_end:
5267     }{
5268       \l_tmpa_tl
5269     }
5270     \end{stex_annotate_env}
5271   }
5272 }

```

\stexpatchassertion

```

5273
5274 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5275   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5276     (\sassertiontitle)
5277   }~}
5278 }
5279 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5280
5281 \newcommand\stexpatchassertion[3] [] {
5282   \str_set:Nx \l_tmpa_str{ #1 }
5283   \str_if_empty:NTF \l_tmpa_str {
5284     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5285     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5286   }{
5287     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5288     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5289   }
5290 }

```

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

\inlineass inline:

```

5291 \keys_define:nn {stex / inlineass }{
5292   type      .str_set_x:N = \sassertiontype,
5293   id        .str_set_x:N = \sassertionid,
5294   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5295   name      .str_set_x:N = \sassertionname

```

```

5296 }
5297 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5298   \str_clear:N \sassertiontype
5299   \str_clear:N \sassertionid
5300   \str_clear:N \sassertionname
5301   \clist_clear:N \l__stex_statements_sassertion_for_clist
5302   \keys_set:nn { stex / inlineass }{ #1 }
5303 }
5304 \NewDocumentCommand \inlineass { 0{} m } {
5305   \begingroup
5306   \stex_reactivate_macro:N \premise
5307   \stex_reactivate_macro:N \conclusion
5308   \__stex_statements_inlineass_args:n{ #1 }
5309   \str_if_empty:NTF \sassertionid {
5310     \str_if_empty:NF \sassertionname {
5311       \stex_ref_new_doc_target:n {}
5312     }
5313   } {
5314     \stex_ref_new_doc_target:n \sassertionid
5315   }
5316
5317   \stex_if_smsmode:TF{
5318     \str_if_empty:NF \sassertionname {
5319       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5320       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5321     }
5322   }{
5323     \seq_clear:N \l_tmpa_seq
5324     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5325       \tl_if_empty:nF{ ##1 }{
5326         \stex_get_symbol:n { ##1 }
5327         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5328           \l_stex_get_symbol_uri_str
5329         }
5330       }
5331     }
5332     \exp_args:Nnx
5333     \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5334       \str_if_empty:NF \sassertiontype {
5335         \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5336       }
5337       #2
5338       \str_if_empty:NF \sassertionname {
5339         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5340         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5341         \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5342       }
5343     }
5344   }
5345   \endgroup
5346   \stex_smsmode_do:
5347 }

```

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

32.3 Examples

sexample

```

5348
5349 \keys_define:nn {stex / sexample }{
5350   type      .str_set_x:N = \exampletype,
5351   id        .str_set_x:N = \sexampleid,
5352   title     .tl_set:N     = \sexampletile,
5353   name      .str_set_x:N = \sexamplename ,
5354   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5355 }
5356 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5357   \str_clear:N \sexampletype
5358   \str_clear:N \sexampleid
5359   \str_clear:N \sexamplename
5360   \tl_clear:N \sexampletile
5361   \clist_clear:N \l__stex_statements_sexample_for_clist
5362   \keys_set:nn { stex / sexample }{ #1 }
5363 }
5364
5365 \NewDocumentEnvironment{sexample}{0{}}{
5366   \__stex_statements_sexample_args:n{ #1 }
5367   \stex_reactivate_macro:N \premise
5368   \stex_reactivate_macro:N \conclusion
5369   \stex_if_smsmode:F {
5370     \seq_clear:N \l_tmpa_seq
5371     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5372       \tl_if_empty:NF{ ##1 }{
5373         \stex_get_symbol:n { ##1 }
5374         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5375           \l_stex_get_symbol_uri_str
5376         }
5377       }
5378     }
5379     \exp_args:Nnnx
5380     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5381     \str_if_empty:NF \sexampletype {
5382       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{ }
5383     }
5384     \str_if_empty:NF \sexamplename {
5385       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5386     }
5387     \clist_set:Nn \l_tmpa_clist \sexampletype
5388     \tl_clear:N \l_tmpa_tl
5389     \clist_map_inline:Nn \l_tmpa_clist {
5390       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5391         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5392       }
5393     }
5394     \tl_if_empty:NTF \l_tmpa_tl {
5395       \__stex_statements_sexample_start:
5396     }{
5397       \l_tmpa_tl
5398     }

```

```

5399 }
5400 \str_if_empty:NF \sexampleid {
5401   \stex_ref_new_doc_target:n \sexampleid
5402 }
5403 \stex_smsmode_do:
5404 }{
5405   \str_if_empty:NF \sexamplename {
5406     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5407   }
5408   \stex_if_smsmode:F {
5409     \clist_set:Nn \l_tmpa_clist \sexamplotype
5410     \tl_clear:N \l_tmpa_tl
5411     \clist_map_inline:Nn \l_tmpa_clist {
5412       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5413         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5414       }
5415     }
5416     \tl_if_empty:NTF \l_tmpa_tl {
5417       \__stex_statements_sexample_end:
5418     }{
5419       \l_tmpa_tl
5420     }
5421     \end{stex_annotate_env}
5422   }
5423 }

```

\stexpatchexample

```

5424
5425 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5426   \par\noindent\titllemph{Example~\tl_if_empty:NF \sexamplotype {
5427     (\sexamplotype)
5428   }~}
5429 }
5430 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5431
5432 \newcommand\stexpatchexample[3] [] {
5433   \str_set:Nx \l_tmpa_str{ #1 }
5434   \str_if_empty:NTF \l_tmpa_str {
5435     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5436     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5437   }{
5438     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5439     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5440   }
5441 }

```

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

\inlineex inline:

```

5442 \keys_define:nn {stex / inlineex }{
5443   type      .str_set_x:N = \sexamplotype,
5444   id        .str_set_x:N = \sexampleid,
5445   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5446   name      .str_set_x:N = \sexamplename

```

```

5447 }
5448 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5449   \str_clear:N \sexamplotype
5450   \str_clear:N \sexampleid
5451   \str_clear:N \sexamplename
5452   \clist_clear:N \l__stex_statements_sexample_for_clist
5453   \keys_set:nn { stex / inlineex }{ #1 }
5454 }
5455 \NewDocumentCommand \inlineex { 0{ } m } {
5456   \beginngroup
5457   \stex_reactivate_macro:N \premise
5458   \stex_reactivate_macro:N \conclusion
5459   \__stex_statements_inlineex_args:n{ #1 }
5460   \str_if_empty:NF \sexampleid {
5461     \stex_ref_new_doc_target:n \sexampleid
5462   }
5463   \stex_if_smsmode:TF{
5464     \str_if_empty:NF \sexamplename {
5465       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5466     }
5467   }{
5468     \seq_clear:N \l_tmpa_seq
5469     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5470       \tl_if_empty:nF{ ##1 }{
5471         \stex_get_symbol:n { ##1 }
5472         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5473           \l_stex_get_symbol_uri_str
5474         }
5475       }
5476     }
5477     \exp_args:Nnx
5478     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5479       \str_if_empty:NF \sexamplotype {
5480         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5481       }
5482       #2
5483       \str_if_empty:NF \sexamplename {
5484         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5485         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5486       }
5487     }
5488   }
5489   \endgroup
5490   \stex_smsmode_do:
5491 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5492 \keys_define:nn { stex / sparagraph } {
5493   id .str_set_x:N = \sparagraphid ,

```



```

5494 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5495 type .str_set_x:N = \sparagraphtype ,
5496 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5497 from .tl_set:N = \sparagraphfrom ,
5498 to .tl_set:N = \sparagraphto ,
5499 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5500 name .str_set:N = \sparagraphname ,
5501 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5502 }
5503
5504 \cs_new_protected:Nn \stex_sparagraph_args:n {
5505 \tl_clear:N \l_stex_sparagraph_title_tl
5506 \tl_clear:N \sparagraphfrom
5507 \tl_clear:N \sparagraphto
5508 \tl_clear:N \l_stex_sparagraph_start_tl
5509 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5510 \str_clear:N \sparagraphid
5511 \str_clear:N \sparagraphtype
5512 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5513 \str_clear:N \sparagraphname
5514 \keys_set:nn { stex / sparagraph }{ #1 }
5515 }
5516 \newif\if@in@omtext\@in@omtextfalse
5517
5518 \NewDocumentEnvironment {sparagraph} { 0{} } {
5519 \stex_sparagraph_args:n { #1 }
5520 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5521 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5522 }{
5523 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5524 }
5525 \@in@omtexttrue
5526 \stex_if_smsmode:F {
5527 \seq_clear:N \l_tmpa_seq
5528 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5529 \tl_if_empty:NF{ ##1 }{
5530 \stex_get_symbol:n { ##1 }
5531 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5532 \l_stex_get_symbol_uri_str
5533 }
5534 }
5535 }
5536 \exp_args:Nnnx
5537 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5538 \str_if_empty:NF \sparagraphtype {
5539 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5540 }
5541 \str_if_empty:NF \sparagraphfrom {
5542 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5543 }
5544 \str_if_empty:NF \sparagraphto {
5545 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5546 }
5547 \str_if_empty:NF \sparagraphname {

```

```

5548     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5549   }
5550   \clist_set:No \l_tmpa_clist \sparagraphtype
5551   \tl_clear:N \l_tmpa_tl
5552   \clist_map_inline:Nn \sparagraphtype {
5553     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5554       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5555     }
5556   }
5557   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5558   \tl_if_empty:NTF \l_tmpa_tl {
5559     \__stex_statements_sparagraph_start:
5560   }{
5561     \l_tmpa_tl
5562   }
5563 }
5564 \clist_set:No \l_tmpa_clist \sparagraphtype
5565 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5566 {
5567   \stex_reactivate_macro:N \definiendum
5568   \stex_reactivate_macro:N \definame
5569   \stex_reactivate_macro:N \Definame
5570   \stex_reactivate_macro:N \premise
5571   \stex_reactivate_macro:N \definiens
5572 }
5573 \str_if_empty:NTF \sparagraphid {
5574   \str_if_empty:NTF \sparagraphname {
5575     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5576       \stex_ref_new_doc_target:n {}
5577     }
5578   } {
5579     \stex_ref_new_doc_target:n {}
5580   }
5581 } {
5582   \stex_ref_new_doc_target:n \sparagraphid
5583 }
5584 \exp_args:NNx
5585 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5586   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5587     \tl_if_empty:nF{ ##1 }{
5588       \stex_get_symbol:n { ##1 }
5589       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5590     }
5591   }
5592 }
5593 \stex_smsmode_do:
5594 \ignorespacesandpars
5595 }{
5596   \str_if_empty:NF \sparagraphname {
5597     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5598     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5599   }
5600   \stex_if_smsmode:F {
5601     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5602 \tl_clear:N \l_tmpa_tl
5603 \clist_map_inline:Nn \l_tmpa_clist {
5604   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5605     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5606   }
5607 }
5608 \tl_if_empty:NTF \l_tmpa_tl {
5609   \__stex_statements_sparagraph_end:
5610 }{
5611   \l_tmpa_tl
5612 }
5613 \end{stex_annotate_env}
5614 }
5615 }

```

\stexpatchparagraph

```

5616
5617 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5618   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5619     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5620       \titleemph{\l_stex_sparagraph_title_tl}:~
5621     }
5622   }{
5623     \titleemph{\l_stex_sparagraph_start_tl}~
5624   }
5625 }
5626 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5627
5628 \newcommand\stexpatchparagraph[3] [] {
5629   \str_set:Nx \l_tmpa_str{ #1 }
5630   \str_if_empty:NTF \l_tmpa_str {
5631     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5632     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5633   }{
5634     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5635     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5636   }
5637 }
5638
5639 \keys_define:nn { stex / inlinepara } {
5640   id      .str_set_x:N = \sparagraphid ,
5641   type    .str_set_x:N = \sparagraphtype ,
5642   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5643   from    .tl_set:N    = \sparagraphfrom ,
5644   to      .tl_set:N    = \sparagraphto ,
5645   name    .str_set:N   = \sparagraphname
5646 }
5647 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5648   \tl_clear:N \sparagraphfrom
5649   \tl_clear:N \sparagraphto
5650   \str_clear:N \sparagraphid
5651   \str_clear:N \sparagraphtype
5652   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5653   \str_clear:N \sparagraphname

```

```

5654 \keys_set:nn { stex / inlinepara }{ #1 }
5655 }
5656 \NewDocumentCommand \inlinepara { 0{} m } {
5657   \beginingroup
5658   \__stex_statements_inlinepara_args:n{ #1 }
5659   \clist_set:No \l_tmpa_clist \sparagraphtype
5660   \str_if_empty:NTF \sparagraphid {
5661     \str_if_empty:NTF \sparagraphname {
5662       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5663         \stex_ref_new_doc_target:n {}
5664       }
5665     } {
5666       \stex_ref_new_doc_target:n {}
5667     }
5668   } {
5669     \stex_ref_new_doc_target:n \sparagraphid
5670   }
5671   \stex_if_smsmode:TF{
5672     \str_if_empty:NF \sparagraphname {
5673       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5674     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5675   }
5676 }{
5677   \seq_clear:N \l_tmpa_seq
5678   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5679     \tl_if_empty:nF{ ##1 }{
5680       \stex_get_symbol:n { ##1 }
5681       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5682         \l_stex_get_symbol_uri_str
5683       }
5684     }
5685   }
5686   \exp_args:Nnx
5687   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5688     \str_if_empty:NF \sparagraphtype {
5689       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5690     }
5691     \str_if_empty:NF \sparagraphfrom {
5692       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5693     }
5694     \str_if_empty:NF \sparagraphto {
5695       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5696     }
5697     \str_if_empty:NF \sparagraphname {
5698       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5699     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5700     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5701   }
5702   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5703     \clist_map_inline:Nn \l_tmpa_seq {
5704       \stex_ref_new_sym_target:n {##1}
5705     }
5706   }
5707   #2

```

```

5708     }
5709   }
5710   \endgroup
5711   \stex_smsmode_do:
5712 }
5713
(End definition for \stexpatchparagraph. This function is documented on page 47.)
5714 \</package>

```

Chapter 33

The Implementation

```
5715 <*package>
5716 <@@=stex_sproof>
5717
5718 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5719
```

33.1 Proofs

We first define some keys for the proof environment.

```
5720 \keys_define:nn { stex / spf } {
5721   id          .str_set_x:N = \spfid,
5722   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5723   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5724   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5725   type        .str_set_x:N = \spftype,
5726   title       .tl_set:N    = \spftitle,
5727   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5728   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5729   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5730 }
5731 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5732   \str_clear:N \spfid
5733   \tl_clear:N \l__stex_sproof_spf_for_tl
5734   \tl_clear:N \l__stex_sproof_spf_from_tl
5735   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5736   \str_clear:N \spftype
5737   \tl_clear:N \spftitle
5738   \tl_clear:N \l__stex_sproof_spf_continues_tl
5739   \tl_clear:N \l__stex_sproof_spf_functions_tl
5740   \tl_clear:N \l__stex_sproof_spf_method_tl
5741   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5742   \keys_set:nn { stex / spf }{ #1 }
5743 }
```

\c__stex_sproof_flow_str

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

```
5744 \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.

```

5745 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5746 \cs_new_protected:Npn \sproofnumber {
5747   \int_set:Nn \l_tmpa_int {1}
5748   \bool_while_do:nn {
5749     \int_compare_p:nNn {
5750       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5751     } > 0
5752   }{
5753     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5754     \int_incr:N \l_tmpa_int
5755   }
5756 }
5757 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5758   \int_set:Nn \l_tmpa_int {1}
5759   \bool_while_do:nn {
5760     \int_compare_p:nNn {
5761       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5762     } > 0
5763   }{
5764     \int_incr:N \l_tmpa_int
5765   }
5766   \int_compare:nNnF \l_tmpa_int = 1 {
5767     \int_decr:N \l_tmpa_int
5768   }
5769   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5770     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5771   }
5772 }
5773
5774 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5775   \int_set:Nn \l_tmpa_int {1}
5776   \bool_while_do:nn {
5777     \int_compare_p:nNn {
5778       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5779     } > 0
5780   }{
5781     \int_incr:N \l_tmpa_int
5782   }
5783   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5784 }
5785
5786 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5787   \int_set:Nn \l_tmpa_int {1}
5788   \bool_while_do:nn {

```

```

5789 \int_compare_p:nNn {
5790 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5791 } > 0
5792 }{
5793 \int_incr:N \l_tmpa_int
5794 }
5795 \int_decr:N \l_tmpa_int
5796 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5797 }

```

\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

```

5798 \def\sproof@box{
5799 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5800 }
5801 \def\sproofend{
5802 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5803 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5804 }
5805 }

```

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

spf@*@kw

```

5806 \def\spf@proofsketch@kw{Proof~Sketch}
5807 \def\spf@proof@kw{Proof}
5808 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5809 \AddToHook{begindocument}{
5810 \ltx@ifpackageloaded{babel}{
5811 \makeatletter
5812 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5813 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5814 \input{sproof-ngerman.ldf}
5815 }
5816 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5817 \input{sproof-finnish.ldf}
5818 }
5819 \clist_if_in:NnT \l_tmpa_clist {french}{
5820 \input{sproof-french.ldf}
5821 }
5822 \clist_if_in:NnT \l_tmpa_clist {russian}{
5823 \input{sproof-russian.ldf}
5824 }
5825 \makeatother
5826 }{}
5827 }

```

spfsketch

```

5828 \newcommand\spsketch[2][]{
5829 \begin{group}
5830 \let \premise \stex_proof_premise:

```



```

5831 \__stex_sproof_spf_args:n{#1}
5832 \stex_if_smsmode:TF {
5833   \str_if_empty:NF \spfid {
5834     \stex_ref_new_doc_target:n \spfid
5835   }
5836 }{
5837   \seq_clear:N \l_tmpa_seq
5838   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5839     \tl_if_empty:nF{ ##1 }{
5840       \stex_get_symbol:n { ##1 }
5841       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5842         \l_stex_get_symbol_uri_str
5843       }
5844     }
5845   }
5846   \exp_args:Nnx
5847   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5848     \str_if_empty:NF \spftype {
5849       \stex_annotate_invisible:nnn{type}{\spftype}{
5850     }
5851     \clist_set:No \l_tmpa_clist \spftype
5852     \tl_set:Nn \l_tmpa_tl {
5853       \titleemph{
5854         \tl_if_empty:NTF \spftitle {
5855           \spf@proofsketch@kw
5856         }{
5857           \spftitle
5858         }
5859       }::~
5860     }
5861     \clist_map_inline:Nn \l_tmpa_clist {
5862       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5863         \tl_clear:N \l_tmpa_tl
5864       }
5865     }
5866     \str_if_empty:NF \spfid {
5867       \stex_ref_new_doc_target:n \spfid
5868     }
5869     \l_tmpa_tl #2 \sproofend
5870   }
5871 }
5872 \endgroup
5873 \stex_smsmode_do:
5874 }
5875

```

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

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

```

5876 \newenvironment{spfeq}[2][ ]{
5877   \__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

```

5878 \let \premise \stex_proof_premise:
5879 \stex_if_smsmode:TF {
5880   \str_if_empty:NF \spfid {
5881     \stex_ref_new_doc_target:n \spfid
5882   }
5883 }{
5884   \seq_clear:N \l_tmpa_seq
5885   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5886     \tl_if_empty:nF{ ##1 }{
5887       \stex_get_symbol:n { ##1 }
5888       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5889         \l_stex_get_symbol_uri_str
5890       }
5891     }
5892   }
5893   \exp_args:Nnnx
5894   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5895   \str_if_empty:NF \spftype {
5896     \stex_annotate_invisible:nnn{type}{\spftype}{ }
5897   }
5898
5899   \clist_set:No \l_tmpa_clist \spftype
5900   \tl_clear:N \l_tmpa_tl
5901   \clist_map_inline:Nn \l_tmpa_clist {
5902     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5903       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5904     }
5905     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5906       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5907     }
5908   }
5909   \tl_if_empty:NTF \l_tmpa_tl {
5910     \__stex_sproof_spfeq_start:
5911   }{
5912     \l_tmpa_tl
5913   }{~#2}
5914   \str_if_empty:NF \spfid {
5915     \stex_ref_new_doc_target:n \spfid
5916   }
5917   \begin{displaymath}\begin{array}{rcll}
5918   }
5919   \stex_smsmode_do:
5920 }{
5921   \stex_if_smsmode:F {
5922     \end{array}\end{displaymath}
5923     \clist_set:No \l_tmpa_clist \spftype
5924     \tl_clear:N \l_tmpa_tl
5925     \clist_map_inline:Nn \l_tmpa_clist {
5926       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5927         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5928       }
5929     }
5930     \tl_if_empty:NTF \l_tmpa_tl {
5931       \__stex_sproof_spfeq_end:

```

```

5932   }{
5933     \l_tmpa_tl
5934   }
5935   \end{stex_annotate_env}
5936 }
5937 }
5938
5939 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5940   \titleemph{
5941     \tl_if_empty:NTF \spftitle {
5942       \spf@proof@kw
5943     }{
5944       \spftitle
5945     }
5946   }:
5947 }
5948 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5949
5950 \newcommand\stexpatchspfeq[3] [] {
5951   \str_set:Nx \l_tmpa_str{ #1 }
5952   \str_if_empty:NTF \l_tmpa_str {
5953     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5954     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5955   }{
5956     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5957     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5958   }
5959 }
5960

```

(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.

```

5961 \newenvironment{sproof}[2] []{
5962   \let \premise \stex_proof_premise:
5963   \intarray_gzero:N \l__stex_sproof_counter_intarray
5964   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5965   \__stex_sproof_spf_args:n{#1}
5966   \stex_if_smsmode:TF {
5967     \str_if_empty:NF \spfid {
5968       \stex_ref_new_doc_target:n \spfid
5969     }
5970   }{
5971     \seq_clear:N \l_tmpa_seq
5972     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5973       \tl_if_empty:nF{ ##1 }{
5974         \stex_get_symbol:n { ##1 }
5975         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5976           \l_stex_get_symbol_uri_str
5977         }
5978       }
5979     }

```

```

5980 \exp_args:Nnnx
5981 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5982 \str_if_empty:NF \spftype {
5983   \stex_annotate_invisible:nnn{type}{\spftype}{}}
5984 }
5985
5986 \clist_set:No \l_tmpa_clist \spftype
5987 \tl_clear:N \l_tmpa_tl
5988 \clist_map_inline:Nn \l_tmpa_clist {
5989   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5990     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5991   }
5992   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5993     \tl_set:Nn \l_tmpa_tl {\use:n{}}
5994   }
5995 }
5996 \tl_if_empty:NTF \l_tmpa_tl {
5997   \__stex_sproof_sproof_start:
5998 }{
5999   \l_tmpa_tl
6000 }{~#2}
6001 \str_if_empty:NF \spfid {
6002   \stex_ref_new_doc_target:n \spfid
6003 }
6004 \begin{description}
6005 }
6006 \stex_smsmode_do:
6007 }{
6008   \stex_if_smsmode:F{
6009     \end{description}
6010     \clist_set:No \l_tmpa_clist \spftype
6011     \tl_clear:N \l_tmpa_tl
6012     \clist_map_inline:Nn \l_tmpa_clist {
6013       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6014         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6015       }
6016     }
6017     \tl_if_empty:NTF \l_tmpa_tl {
6018       \__stex_sproof_sproof_end:
6019     }{
6020       \l_tmpa_tl
6021     }
6022     \end{stex_annotate_env}
6023   }
6024 }
6025
6026 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6027   \par\noindent\titleemph{
6028     \tl_if_empty:NTF \spftype {
6029       \spf@proof@kw
6030     }{
6031       \spftype
6032     }
6033   }::

```

```

6034 }
6035 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6036
6037 \newcommand\stexpatchproof[3] [] {
6038   \str_set:Nx \l_tmpa_str{ #1 }
6039   \str_if_empty:NTF \l_tmpa_str {
6040     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6041     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6042   }{
6043     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6044     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6045   }
6046 }

```

\spfidea

```

6047 \newcommand\spfidea[2] []{
6048   \__stex_sproof_spf_args:n{#1}
6049   \titleemph{
6050     \tl_if_empty:NTF \spftype {Proof-Idea}{
6051       \spftype
6052     }:
6053   }~#2
6054   \sproofend
6055 }

```

(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

```

6056 \newenvironment{spfstep}[1] []{
6057   \__stex_sproof_spf_args:n{#1}
6058   \stex_if_smsmode:TF {
6059     \str_if_empty:NF \spfid {
6060       \stex_ref_new_doc_target:n \spfid
6061     }
6062   }{
6063     \@in@omtexttrue
6064     \seq_clear:N \l_tmpa_seq
6065     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6066       \tl_if_empty:nF{ ##1 }{
6067         \stex_get_symbol:n { ##1 }
6068         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6069           \l_stex_get_symbol_uri_str
6070         }
6071       }
6072     }
6073     \exp_args:Nnnx
6074     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6075     \str_if_empty:NF \spftype {
6076       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

6077 }
6078 \clist_set:No \l_tmpa_clist \spftype
6079 \tl_set:Nn \l_tmpa_tl {
6080   \item[\sproofnumber]
6081   \bool_set_true:N \l__stex_sproof_inc_counter_bool
6082 }
6083 \clist_map_inline:Nn \l_tmpa_clist {
6084   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6085     \tl_clear:N \l_tmpa_tl
6086   }
6087 }
6088 \l_tmpa_tl
6089 \tl_if_empty:NF \spftitle {
6090   {(\titleemph{\spftitle})\enspace}
6091 }
6092 \str_if_empty:NF \spfid {
6093   \stex_ref_new_doc_target:n \spfid
6094 }
6095 }
6096 \stex_smsmode_do:
6097 \ignorespacesandpars
6098 }{
6099   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6100     \__stex_sproof_inc_counter:
6101   }
6102   \stex_if_smsmode:F {
6103     \end{stex_annotate_env}
6104   }
6105 }

```

spfcomment

```

6106 \newenvironment{spfcomment}[1][]{
6107   \__stex_sproof_spf_args:n{#1}
6108   \clist_set:No \l_tmpa_clist \spftype
6109   \tl_set:Nn \l_tmpa_tl {
6110     \item[\sproofnumber]
6111     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6112   }
6113   \clist_map_inline:Nn \l_tmpa_clist {
6114     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6115       \tl_clear:N \l_tmpa_tl
6116     }
6117   }
6118   \l_tmpa_tl
6119 }{
6120   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6121     \__stex_sproof_inc_counter:
6122   }
6123 }

```

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.

```

6124 \newenvironment{subproof}[2][]{
6125   \_stex_sproof_spf_args:n{#1}
6126   \stex_if_smsmode:TF{
6127     \str_if_empty:NF \spfid {
6128       \stex_ref_new_doc_target:n \spfid
6129     }
6130   }{
6131     \seq_clear:N \l_tmpa_seq
6132     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6133       \tl_if_empty:nF{ ##1 }{
6134         \stex_get_symbol:n { ##1 }
6135         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6136           \l_stex_get_symbol_uri_str
6137         }
6138       }
6139     }
6140     \exp_args:Nnnx
6141     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6142     \str_if_empty:NF \spftype {
6143       \stex_annotate_invisible:nnn{type}{\spftype}{\}
6144     }
6145
6146     \clist_set:No \l_tmpa_clist \spftype
6147     \tl_set:Nn \l_tmpa_tl {
6148       \item[\sproofnumber]
6149       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6150     }
6151     \clist_map_inline:Nn \l_tmpa_clist {
6152       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6153         \tl_clear:N \l_tmpa_tl
6154       }
6155     }
6156     \l_tmpa_tl
6157     \tl_if_empty:NF \spftitle {
6158       {(\titleemph{\spftitle})\enspace}
6159     }
6160     {\~#2}
6161     \str_if_empty:NF \spfid {
6162       \stex_ref_new_doc_target:n \spfid
6163     }
6164   }
6165   \_stex_sproof_add_counter:
6166   \stex_smsmode_do:
6167 }{
6168   \_stex_sproof_remove_counter:
6169   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6170     \_stex_sproof_inc_counter:
6171   }
6172   \stex_if_smsmode:F{
6173     \end{stex_annotate_env}
6174   }
6175 }

```

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

```

6176 \newenvironment{spfcases}[2] [] {
6177   \tl_if_empty:nTF{#1}{
6178     \begin{subproof}[method=by-cases]{#2}
6179   }{
6180     \begin{subproof}[#1,method=by-cases]{#2}
6181   }
6182 }{
6183   \end{subproof}
6184 }

```

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

```

6185 \newenvironment{spfcase}[2] [] {
6186   \__stex_sproof_spf_args:n{#1}
6187   \stex_if_smsmode:TF {
6188     \str_if_empty:NF \spfid {
6189       \stex_ref_new_doc_target:n \spfid
6190     }
6191   }{
6192     \seq_clear:N \l_tmpa_seq
6193     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6194       \tl_if_empty:nF{ ##1 }{
6195         \stex_get_symbol:n { ##1 }
6196         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6197           \l_stex_get_symbol_uri_str
6198         }
6199       }
6200     }
6201     \exp_args:Nnnx
6202     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6203     \str_if_empty:NF \spftype {
6204       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6205     }
6206     \clist_set:No \l_tmpa_clist \spftype
6207     \tl_set:Nn \l_tmpa_tl {
6208       \item[\sproofnumber]
6209       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6210     }
6211     \clist_map_inline:Nn \l_tmpa_clist {
6212       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6213         \tl_clear:N \l_tmpa_tl
6214       }
6215     }
6216     \l_tmpa_tl
6217     \tl_if_empty:nF{#2}{
6218       \titleemph{#2}:~
6219     }
6220   }
6221   \__stex_sproof_add_counter:
6222   \stex_smsmode_do:
6223 }{
6224   \__stex_sproof_remove_counter:
6225   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6226     \__stex_sproof_inc_counter:

```



```

6227 }
6228 \stex_if_smsmode:F{
6229   \clist_set:No \l_tmpa_clist \spftype
6230   \tl_set:Nn \l_tmpa_tl{\sproofend}
6231   \clist_map_inline:Nn \l_tmpa_clist {
6232     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6233       \tl_clear:N \l_tmpa_tl
6234     }
6235   }
6236   \l_tmpa_tl
6237   \end{stex_annotate_env}
6238 }
6239 }

```

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

```

6240 \newcommand\spfcasesketch[3] [] {
6241   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6242 }

```

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.

```

6243 \keys_define:nn { stex / just }{
6244   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6245   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6246   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6247   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6248 }

```

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

\spfjust

```

6249 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6250 \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.

```

6251 \newcommand\justarg[2] [] {#2}
6252 \</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

```
6253 <*package>
6254
6255 %%%%%%%%%% others.dtx %%%%%%%%%%
6256
6257 <@@=stex_others>
        Warnings and error messages
6258 % None

\MSC Math subject classifier

6259 \NewDocumentCommand \MSC {m} {
6260 % TODO
6261 }

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

6262 \@ifpackageloaded{tikzinput}{
6263 \RequirePackage{stex-tikzinput}
6264 }{}
6265
6266 \bool_if:NT \c_stex_persist_mode_bool {
6267 \input{\jobname.sms}
6268 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6269 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6270 \l_tmpa_str
6271 \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
6272 \c_stex_mathhub_main_manifest_prop
6273 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6274 }
6275 }

6276 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6277 <*package>
6278 <@@=stex_modules>
6279
6280 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6281
6282 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6283 \begingroup
6284 \stex_module_setup:nn{
6285   ns=\c_stex_metatheory_ns_str,
6286   meta=NONE
6287 }{Metatheory}
6288 \stex_reactivate_macro:N \symdecl
6289 \stex_reactivate_macro:N \notation
6290 \stex_reactivate_macro:N \symdef
6291 \ExplSyntaxOff
6292 \csname stex_suppress_html:n\endcsname{
6293   % is-a (a:A, a \in A, a is an A, etc.)
6294   \symdecl{isa}[args=ai]
6295   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6296   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6297   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6298
6299   % bind (\forall, \Pi, \lambda etc.)
6300   \symdecl{bind}[args=Bi]
6301   \notation{bind}{forall}{\comp\forall #1.;#2}{##1 \comp, ##2}
6302   \notation{bind}{Pi}{\comp\prod_{#1}#2}{##1 \comp, ##2}
6303   \notation{bind}{depfun}{\comp( #1 \comp{} \;\to\; ) #2}{##1 \comp, ##2}
6304
6305   % implicit bind
6306   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6307
6308   % dummy variable
6309   \symdecl{dummyvar}
6310   \notation{dummyvar}[underscore]{\comp\_}
6311   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6312 \notation{dummyvar}[dash]{\comp{\rm --}}
6313
6314 %fromto (function space, Hom-set, implication etc.)
6315 \symdecl{fromto}[args=ai]
6316 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6317 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6318
6319 % mapto (lambda etc.)
6320 \symdecl{mapto}[args=Bi]
6321 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6322 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6323 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6324
6325 % function/operator application
6326 \symdecl{apply}[args=ia]
6327 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6328 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6329
6330 % collection of propositions/booleans/truth values
6331 \symdecl{prop}[name=proposition]
6332 \notation{prop}[prop]{\comp{\rm prop}}
6333 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6334
6335 \symdecl{judgmentholds}[args=1]
6336 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6337
6338 % sequences
6339 \symdecl{seqtype}[args=1]
6340 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6341
6342 \symdecl{seqexpr}[args=a]
6343 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6344
6345 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6346 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6347 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6348 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6349 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6350 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6351 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6352 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6353 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6354
6355 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6356 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6357
6358 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6359 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6360 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6361
6362 % letin ("let", local definitions, variable substitution)
6363 \symdecl{letin}[args=bii]
6364 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
6365 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6366 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6367
6368 % structures
6369 \symdecl*{module-type}[args=1]
6370 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6371 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6372 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6373
6374 % objects
6375 \symdecl{object}
6376 \notation{object}{\comp{\mathtt{OBJECT}}}
6377
6378 }
6379
6380 % The following are abbreviations in the sTeX corpus that are left over from earlier
6381 % developments. They will eventually be phased out.
6382
6383 \ExplSyntaxOn
6384 \stex_add_to_current_module:n{
6385   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6386   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6387   \def\livar{\csname sequence-index\endcsname[li]}
6388   \def\uivar{\csname sequence-index\endcsname[ui]}
6389   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6390   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6391 }
6392 \__stex_modules_end_module:
6393 \endgroup
6394 \endpackage

```

Chapter 36

Tikzinput Implementation

```
6395 <@@=tikzinput>
6396 <*package>
6397
6398 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6399
6400 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6401 \RequirePackage{l3keys2e}
6402
6403 \keys_define:nn { tikzinput } {
6404   image .bool_set:N = \c_tikzinput_image_bool,
6405   image .default:n = false ,
6406   unknown .code:n = {}
6407 }
6408
6409 \ProcessKeysOptions { tikzinput }
6410
6411 \bool_if:NTF \c_tikzinput_image_bool {
6412   \RequirePackage{graphicx}
6413
6414   \providecommand\usetikzlibrary[]{}
6415   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6416 }{
6417   \RequirePackage{tikz}
6418   \RequirePackage{standalone}
6419
6420   \newcommand \tikzinput [2] [] {
6421     \setkeys{Gin}{#1}
6422     \ifx \Gin@ewidth \Gin@exclamation
6423       \ifx \Gin@eheight \Gin@exclamation
6424         \input { #2 }
6425       \else
6426         \resizebox{!}{ \Gin@eheight }{
6427           \input { #2 }
6428         }
6429       \fi
6430     \else
6431       \ifx \Gin@eheight \Gin@exclamation
6432         \resizebox{ \Gin@ewidth }{!}{
```

```

6433         \input { #2 }
6434     }
6435     \else
6436         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6437             \input { #2 }
6438         }
6439     \fi
6440 \fi
6441 }
6442 }
6443
6444 \newcommand \ctikzinput [2] [] {
6445     \begin{center}
6446         \tikzinput [#1] {#2}
6447     \end{center}
6448 }
6449
6450 \@ifpackageloaded{stex}{
6451     \RequirePackage{stex-tikzinput}
6452 }{}
6453
6454 \</package>
6455 \<stex>
6456 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6457 \RequirePackage{stex}
6458 \RequirePackage{tikzinput}
6459
6460 \newcommand\mhtikzinput[2] []{%
6461     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6462     \stex_in_repository:nn\Gin@mhrepos{
6463         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6464     }
6465 }
6466 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6467
6468 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6469     \pgfkeys@spdef\pgf@temp{#1}
6470     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6471     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6472     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6473     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6474     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6475     \catcode'\@=11
6476     \catcode'\|=12
6477     \catcode'\$=3
6478     \pgfutil@InputIfFileExists{#2}{-}{-}
6479     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6480     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6481     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6482 }
6483
6484
6485 \newcommand\libusetikzlibrary[1]{

```

```

6486 \prop_if_exist:NF \l_stex_current_repository_prop {
6487   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6488 }
6489 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6490   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6491 }
6492 \seq_clear:N \l__tikzinput_libinput_files_seq
6493 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6494 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6495
6496 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6497   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6498   \IfFileExists{ \l_tmpa_str }{
6499     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6500   }{}
6501   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6502   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6503 }
6504
6505 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6506 \IfFileExists{ \l_tmpa_str }{
6507   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6508 }{}
6509
6510 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6511   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6512 }{
6513   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6514     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6515       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6516     }
6517   }{
6518     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6519   }
6520 }
6521 }
6522 </stex>

```

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

Chapter 37

document-structure.sty Implementation

```
6523 \*package>
6524 \@@=document_structure>
6525 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6526 \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).

```
6527
6528 \keys_define:nn{ document-structure }{
6529   class      .str_set_x:N = \c_document_structure_class_str,
6530   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6531   unknown    .code:n      = {
6532     \PassOptionsToClass{\CurrentOption}{stex}
6533     \PassOptionsToClass{\CurrentOption}{tikzinput}
6534   }
6535   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6536 }
6537 \ProcessKeysOptions{ document-structure }
6538 \str_if_empty:NT \c_document_structure_class_str {
6539   \str_set:Nn \c_document_structure_class_str {article}
6540 }
6541 \str_if_empty:NT \c_document_structure_topsect_str {
6542   \str_set:Nn \c_document_structure_topsect_str {section}
6543 }
```

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

```
6544 \RequirePackage{xspace}
6545 \RequirePackage{comment}
6546 \RequirePackage{stex}
6547 \AddToHook{begindocument}{}
```

```

6548 \ltx@ifpackageloaded{babel}{
6549   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6550   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6551     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6552   }
6553 }{}
6554 }

```

`\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.

```

6555 \int_new:N \l_document_structure_section_level_int
6556 \str_case:NnF \c_document_structure_topsect_str {
6557   {part}}{
6558     \int_set:Nn \l_document_structure_section_level_int {0}
6559   }
6560   {chapter}{
6561     \int_set:Nn \l_document_structure_section_level_int {1}
6562   }
6563 }{
6564   \str_case:NnF \c_document_structure_class_str {
6565     {book}{
6566       \int_set:Nn \l_document_structure_section_level_int {0}
6567     }
6568     {report}{
6569       \int_set:Nn \l_document_structure_section_level_int {0}
6570     }
6571   }{
6572     \int_set:Nn \l_document_structure_section_level_int {2}
6573   }
6574 }

```

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

```

6575 \def\current@section@level{document}%
6576 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6577 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6578 \cs_new_protected:Npn \skipfragment {

```

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

```

6579 \ifcase\l_document_structure_section_level_int
6580 \or\stepcounter{part}
6581 \or\stepcounter{chapter}
6582 \or\stepcounter{section}
6583 \or\stepcounter{subsection}
6584 \or\stepcounter{subsubsection}
6585 \or\stepcounter{paragraph}
6586 \or\stepcounter{subparagraph}
6587 \fi
6588 }

```

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

blindfragment

```

6589 \newcommand\at@begin@blindsfragment[1]{
6590 \newenvironment{blindfragment}
6591 {
6592 \int_incr:N\l_document_structure_section_level_int
6593 \at@begin@blindsfragment\l_document_structure_section_level_int
6594 }{}

```

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

```

6595 \newcommand\sfragment@nonum[2]{
6596 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6597 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6598 }

```

(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.

```

6599 \newcommand\sfragment@num[2]{
6600 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6601 \@nameuse{#1}{#2}
6602 }{
6603 \cs_if_exist:NTF\rdmeta@sectioning{
6604 \@nameuse{rdmeta@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6605 }{
6606 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6607 }
6608 }
6609 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6610 }

```

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

sfragment

```

6611 \keys_define:nn { document-structure / sfragment }{
6612 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6613 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6614 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6615 contributors   .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6616 srccite        .tl_set:N    = \l__document_structure_sfragment_srccite_tl,
6617 type          .tl_set:N    = \l__document_structure_sfragment_type_tl,
6618 short         .tl_set:N    = \l__document_structure_sfragment_short_tl,
6619 display       .tl_set:N    = \l__document_structure_sfragment_display_tl,
6620 intro         .tl_set:N    = \l__document_structure_sfragment_intro_tl,
6621 imports       .tl_set:N    = \l__document_structure_sfragment_imports_tl,
6622 loadmodules   .bool_set:N  = \l__document_structure_sfragment_loadmodules_bool
6623 }
6624 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6625   \str_clear:N \l__document_structure_sfragment_id_str
6626   \str_clear:N \l__document_structure_sfragment_date_str
6627   \clist_clear:N \l__document_structure_sfragment_creators_clist
6628   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6629   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6630   \tl_clear:N \l__document_structure_sfragment_type_tl
6631   \tl_clear:N \l__document_structure_sfragment_short_tl
6632   \tl_clear:N \l__document_structure_sfragment_display_tl
6633   \tl_clear:N \l__document_structure_sfragment_imports_tl
6634   \tl_clear:N \l__document_structure_sfragment_intro_tl
6635   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6636   \keys_set:nn { document-structure / sfragment } { #1 }
6637 }

```

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.

```

6638 \newif@if@mainmatter\@mainmattertrue
6639 \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.

```

6640 \keys_define:nn { document-structure / sectioning }{
6641   name      .str_set_x:N = \l__document_structure_sect_name_str ,
6642   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
6643   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6644   clear     .default:n   = {true} ,
6645   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6646   num       .default:n   = {true}
6647 }
6648 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6649   \str_clear:N \l__document_structure_sect_name_str
6650   \str_clear:N \l__document_structure_sect_ref_str
6651   \bool_set_false:N \l__document_structure_sect_clear_bool
6652   \bool_set_false:N \l__document_structure_sect_num_bool
6653   \keys_set:nn { document-structure / sectioning } { #1 }
6654 }
6655 \newcommand\omdoc@sectioning[3][]{
6656   \l__document_structure_sect_args:n {#1 }
6657   \let\omdoc@sect@name\l__document_structure_sect_name_str
6658   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6659   \if@mainmatter% numbering not overridden by frontmatter, etc.
6660     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6661     \sfragment@num{#2}{#3}
6662   }{
6663     \sfragment@nonum{#2}{#3}
6664   }
6665   \def\current@section@level{\omdoc@sect@name}
6666 \else
6667   \sfragment@nonum{#2}{#3}
6668 \fi
6669 }% 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.

```

6670 \newcommand\sfragment@redefine@addtocontents[1]{%
6671   %\edef\__document_structureimport{#1}%
6672   %\@for\@I:=\__document_structureimport\do{%
6673     %\edef\@path{\csname module@\@I @path\endcsname}%
6674     %\@ifundefined{tf@toc}\relax%
6675     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6676     %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6677     %\def\addcontentsline##1##2##3{%
6678       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6679     %\else% hyperref.sty not loaded
6680     %\def\addcontentsline##1##2##3{%
6681       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6682     %\fi
6683   }% 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.

```

6684 \newenvironment{sfragment}[2] []% keys, title
6685 {
6686   \__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.

```

6687   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6688
6689   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6690     \sfragment@redefine@addtocontents{
6691       %\@ifundefined{module@id}\used@modules%
6692       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6693     }
6694   }

```

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

```

6695
6696   \stex_document_title:n { #2 }
6697
6698   \int_incr:N\l__document_structure_section_level_int
6699   \ifcase\l__document_structure_section_level_int
6700     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6701     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6702 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6703 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6704 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6705 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#1}
6706 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#1}
6707 \fi
6708 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6709 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6710 \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6711 }
6712 }% for customization
6713 {}

```

and finally, we localize the sections

```

6714 \newcommand\omdoc@part@kw{Part}
6715 \newcommand\omdoc@chapter@kw{Chapter}
6716 \newcommand\omdoc@section@kw{Section}
6717 \newcommand\omdoc@subsection@kw{Subsection}
6718 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6719 \newcommand\omdoc@paragraph@kw{paragraph}
6720 \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`

```

6721 \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).

```

6722 \cs_if_exist:NTF\frontmatter{
6723 \let\__document_structure_orig_frontmatter\frontmatter
6724 \let\frontmatter\relax
6725 }{
6726 \tl_set:Nn\__document_structure_orig_frontmatter{
6727 \clearpage
6728 \@mainmatterfalse
6729 \pagenumbering{roman}
6730 }
6731 }
6732 \cs_if_exist:NTF\backmatter{
6733 \let\__document_structure_orig_backmatter\backmatter
6734 \let\backmatter\relax
6735 }{
6736 \tl_set:Nn\__document_structure_orig_backmatter{
6737 \clearpage
6738 \@mainmatterfalse

```

```

6739     \pagenumbering{roman}
6740   }
6741 }

```

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.

```

6742 \newenvironment{frontmatter}{
6743   \_document_structure_orig_frontmatter
6744 }{
6745   \cs_if_exist:NTF\mainmatter{
6746     \mainmatter
6747   }{
6748     \clearpage
6749     \@mainmattertrue
6750     \pagenumbering{arabic}
6751   }
6752 }

```

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

```

6753 \newenvironment{backmatter}{
6754   \_document_structure_orig_backmatter
6755 }{
6756   \cs_if_exist:NTF\mainmatter{
6757     \mainmatter
6758   }{
6759     \clearpage
6760     \@mainmattertrue
6761     \pagenumbering{arabic}
6762   }
6763 }

```

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

```

6764 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6765 \def \c__document_structure_document_str{document}
6766 \newcommand\afterprematurestop{}
6767 \def\prematurestop@endsfragment{
6768   \unless\ifx\@currenvir\c__document_structure_document_str
6769     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6770       \expandafter\prematurestop@endsfragment
6771     }
6772   }
6773 \providecommand\prematurestop{
6774   \message{Stopping~sTeX~processing~prematurely}
6775   \prematurestop@endsfragment
6776   \afterprematurestop
6777   \end{document}
6778 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
6779 \RequirePackage{etoolbox}
6780 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6781 \newrobustcmd\useSGvar[1]{%
6782   \@ifundefined{sTeX@Gvar@#1}
6783   {\PackageError{document-structure}
6784     {The sTeX Global variable #1 is undefined}
6785     {set it with \protect\setSGvar}}
6786   \@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.

```
6787 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6788   \@ifundefined{sTeX@Gvar@#1}
6789   {\PackageError{document-structure}
6790     {The sTeX Global variable #1 is undefined}
6791     {set it with \protect\setSGvar}}
6792   {\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.

```
6793 \*cls)
6794 \@@=notesslides}
6795 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6796 \RequirePackage{13keys2e}
6797
6798 \keys_define:nn{notesslides / cls}{
6799   class .str_set_x:N = \c__notesslides_class_str,
6800   notes .bool_set:N = \c__notesslides_notes_bool ,
6801   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6802   docopt .str_set_x:N = \c__notesslides_docopt_str,
6803   unknown .code:n = {
6804     \PassOptionsToPackage{\CurrentOption}{document-structure}
6805     \PassOptionsToClass{\CurrentOption}{beamer}
6806     \PassOptionsToPackage{\CurrentOption}{notesslides}
6807     \PassOptionsToPackage{\CurrentOption}{stex}
6808   }
6809 }
6810 \ProcessKeysOptions{ notesslides / cls }
6811
6812 \str_if_empty:NF \c__notesslides_class_str {
6813   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6814 }
6815
6816 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6817   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6818 }
6819 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6820   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6821 }
6822
6823 \RequirePackage{stex}
```

```

6824 \stex_html_backend:T {
6825   \bool_set_true:N\c__notesslides_notes_bool
6826 }
6827
6828 \bool_if:NTF \c__notesslides_notes_bool {
6829   \PassOptionsToPackage{notes=true}{notesslides}
6830 }{
6831   \PassOptionsToPackage{notes=false}{notesslides}
6832 }
6833 \</cls>

```

now we do the same for the notesslides package.

```

6834 \*package>
6835 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6836 \RequirePackage{13keys2e}
6837
6838 \keys_define:nn{notesslides / pkg}{
6839   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6840   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6841   notes            .bool_set:N = \c__notesslides_notes_bool ,
6842   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6843   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6844   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6845   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6846   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
6847   unknown          .code:n      = {
6848     \PassOptionsToClass{\CurrentOption}{stex}
6849     \PassOptionsToClass{\CurrentOption}{tikzinput}
6850   }
6851 }
6852 \ProcessKeysOptions{ notesslides / pkg }
6853
6854 \RequirePackage{stex}
6855 \stex_html_backend:T {
6856   \bool_set_true:N\c__notesslides_notes_bool
6857 }
6858
6859 \newif\ifnotes
6860 \bool_if:NTF \c__notesslides_notes_bool {
6861   \notesttrue
6862 }{
6863   \notesfalse
6864 }
6865

```

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

```

6866 \str_if_empty:NTF \c__notesslides_topsect_str {
6867   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6868 }{
6869   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6870 }
6871 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}
6872 \</package>

```

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

```

6873 <*cls>
6874 \bool_if:NTF \c__notesslides_notes_bool {
6875   \str_if_empty:NT \c__notesslides_class_str {
6876     \str_set:Nn \c__notesslides_class_str {article}
6877   }
6878   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6879     {\c__notesslides_class_str}
6880 }{
6881   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6882   \newcounter{Item}
6883   \newcounter{paragraph}
6884   \newcounter{subparagraph}
6885   \newcounter{Hfootnote}
6886 }
6887 \RequirePackage{document-structure}

```

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

```

6888 \RequirePackage{notesslides}
6889 </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.

```

6890 <*package>
6891 \bool_if:NT \c__notesslides_notes_bool {
6892   \RequirePackage{a4wide}
6893   \RequirePackage{marginnote}
6894   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6895   \RequirePackage{mdframed}
6896   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6897   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6898 }
6899 \RequirePackage{stex-tikzinput}
6900 \RequirePackage{etoolbox}
6901 \RequirePackage{amssymb}
6902 \RequirePackage{amsmath}
6903 \RequirePackage{comment}
6904 \RequirePackage{textcomp}
6905 \RequirePackage{url}
6906 \RequirePackage{graphicx}
6907 \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

```

6908 \bool_if:NT \c__notesslides_notes_bool {
6909   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6910 }
6911
6912
6913 \NewDocumentCommand \libusetheme {0{} m} {
6914   \bool_if:NTF \c__notesslides_notes_bool {
6915     \libusepackage[#1]{beamernotestheme#2}
6916   }{
6917     \libusepackage[#1]{beamertheme#2}
6918   }
6919 }

```

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

```

6920 \newcounter{slide}
6921 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6922 \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.

```

6923 \bool_if:NTF \c__notesslides_notes_bool {
6924   \renewenvironment{note}{\ignorespaces}{}
6925 }{
6926   \excludcomment{note}
6927 }

```

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.

```

6928 \bool_if:NT \c__notesslides_notes_bool {
6929   \newlength{\slideframewidth}
6930   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6931 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6932   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6933     \bool_set_true:N #1
6934   }{
6935     \bool_set_false:N #1
6936   }
6937 }
6938 \keys_define:nn{notesslides / frame}{
6939   label .str_set_x:N = \l__notesslides_frame_label_str,
6940   allowframebreaks .code:n = {
6941     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6942   },
6943   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.

```

6944     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6945 },
6946 fragile .code:n = {
6947     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6948 },
6949 shrink .code:n = {
6950     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6951 },
6952 squeeze .code:n = {
6953     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6954 },
6955 t .code:n = {
6956     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6957 },
6958 unknown .code:n = {}
6959 }
6960 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6961     \str\_clear:N \l\_notesslides\_frame\_label\_str
6962     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6963     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6964     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6965     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6966     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6967     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6968     \keys\_set:nn { notesslides / frame }{ #1 }
6969 }

```

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

```

6970 \renewenvironment{frame}[1][]{
6971     \_notesslides\_frame\_args:n{#1}
6972     \sffamily
6973     \stepcounter{slide}
6974     \def\@currentlabel{\theslide}
6975     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6976         \label{\l\_notesslides\_frame\_label\_str}
6977     }

```

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

```

6978     \def\itemize@level{outer}
6979     \def\itemize@outer{outer}
6980     \def\itemize@inner{inner}
6981     \renewcommand\newpage{\addtocounter{framenum}{1}}
6982     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6983     \renewenvironment{itemize}{
6984         \ifx\itemize@level\itemize@outer
6985             \def\itemize@label{\$ \rhd \$}
6986         \fi
6987         \ifx\itemize@level\itemize@inner
6988             \def\itemize@label{\$ \scriptstyle \rhd \$}
6989         \fi
6990         \begin{list}
6991             {\itemize@label}
6992             {\setlength{\labelsep}{.3em}
6993              \setlength{\labelwidth}{.5em}
6994              \setlength{\leftmargin}{1.5em}

```

```

6995     }
6996     \edef\itemize@level{\itemize@inner}
6997   }{
6998     \end{list}
6999   }

```

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

```

7000     \stex_html_backend:TF {
7001       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7002     }{
7003       \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7004     }
7005   }{
7006     \stex_html_backend:TF {
7007       \miko@slidelabel\egroup\end{stex_annotate_env}
7008     }\medskip\miko@slidelabel\end{mdframed}}
7009   }

```

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

```

\frametitle
7010   \renewcommand{\frametitle}[1]{
7011     \stex_document_title:n { #1 }
7012     {\Large\bf\sf\color{blue}{#1}}\medskip
7013   }
7014 }

```

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

EdN:19

```

\pause 19
7015 \bool_if:NT \c__notesslides_notes_bool {
7016   \newcommand\pause{}
7017 }

```

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

```

nparagraph
7018 \bool_if:NTF \c__notesslides_notes_bool {
7019   \newenvironment{nparagraph}[1][\begin{sparagraph}[#1]]{\end{sparagraph}}
7020 }{
7021   \excludecomment{nparagraph}
7022 }

```

```

nfragment
7023 \bool_if:NTF \c__notesslides_notes_bool {
7024   \newenvironment{nfragment}[2][\begin{sfragment}[#1]{#2}]{\end{sfragment}}
7025 }{
7026   \excludecomment{nfragment}
7027 }

```

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

ndefinition

```
7028 \bool_if:NTF \c__notesslides_notes_bool {
7029   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7030 }{
7031   \excludecomment{ndefinition}
7032 }
```

nassertion

```
7033 \bool_if:NTF \c__notesslides_notes_bool {
7034   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]\end{sassertion}}
7035 }{
7036   \excludecomment{nassertion}
7037 }
```

nsproof

```
7038 \bool_if:NTF \c__notesslides_notes_bool {
7039   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}\end{sproof}}
7040 }{
7041   \excludecomment{nproof}
7042 }
```

nexample

```
7043 \bool_if:NTF \c__notesslides_notes_bool {
7044   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]\end{sexample}}
7045 }{
7046   \excludecomment{nexample}
7047 }
```

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

```
7048 \def\inputref@preskip{\smallskip}
7049 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7050 \let\orig@inputref\inputref
7051 \def\inputref{\@ifstar\ninputref\orig@inputref}
7052 \newcommand\ninputref[2] [] {
7053   \bool_if:NT \c__notesslides_notes_bool {
7054     \orig@inputref[#1]{#2}
7055   }
7056 }
```

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

38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

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

```

7057 \newlength{\slideloheight}
7058
7059 \bool_if:NTF \c_notesslides_notes_bool {
7060   \setlength{\slideloheight}{.4cm}
7061 }{
7062   \setlength{\slideloheight}{1cm}
7063 }
7064 \newsavebox{\slidelo}
7065 \sbox{\slidelo}{\TeX}
7066 \newrobustcmd{\setslidelo}[1]{
7067   \sbox{\slidelo}{\includegraphics[height=\slideloheight]{#1}}
7068 }

```

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

\setsource `\source` stores the writer's name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name.

```

7069 \def\source{Michael Kohlhase}% customize locally
7070 \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.

```

7071 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7072 \newsavebox{\cclogo}
7073 \sbox{\cclogo}{\includegraphics[height=\slideloheight]{stex-cc_somerights}}
7074 \newif\ifcchref\cchreffalse
7075 \AtBeginDocument{
7076   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7077 }
7078 \def\licensing{
7079   \ifcchref
7080     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
7081   \else
7082     {\usebox{\cclogo}}
7083   \fi
7084 }
7085 \newrobustcmd{\setlicensing}[2][]{
7086   \def\@url{#1}
7087   \sbox{\cclogo}{\includegraphics[height=\slideloheight]{#2}}
7088   \ifx\@url\@empty
7089     \def\licensing{{\usebox{\cclogo}}}
7090   \else
7091     \def\licensing{
7092       \ifcchref
7093         \href{#1}{\usebox{\cclogo}}
7094       \else
7095         {\usebox{\cclogo}}
7096       \fi

```



```

7097     }
7098     \fi
7099 }

```

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

EdN:20

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

```

7100 \newrobustcmd\miko@slidelabel{
7101   \vbox to \slidelogoheight{
7102     \vss\hbox to \slidewidth
7103     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7104   }
7105 }

```

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

38.4 Frame Images

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

```

7106 \def\Gin@mhrepos{}
7107 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7108 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7109 \newrobustcmd\frameimage[2][]{
7110   \stepcounter{slide}
7111   \bool_if:NT \c__notesslides_frameimages_bool {
7112     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7113     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7114     \begin{center}
7115       \bool_if:NTF \c__notesslides_fiboxed_bool {
7116         \fbox{
7117           \ifx\Gin@ewidth\@empty
7118             \ifx\Gin@mhrepos\@empty
7119               \mhgraphics[width=\slidewidth,#1]{#2}
7120             \else
7121               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7122             \fi
7123           \else% Gin@ewidth empty
7124             \ifx\Gin@mhrepos\@empty
7125               \mhgraphics[#1]{#2}
7126             \else
7127               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7128             \fi
7129           \fi% Gin@ewidth empty
7130         }
7131       }{
7132         \ifx\Gin@ewidth\@empty
7133           \ifx\Gin@mhrepos\@empty
7134             \mhgraphics[width=\slidewidth,#1]{#2}
7135           \else
7136             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7137           \fi

```

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

```

7138         \ifx\Gin@mhrepos\@empty
7139             \mhgraphics[#1]{#2}
7140         \else
7141             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7142         \fi
7143     \fi% Gin@ewidth empty
7144 }
7145 \end{center}
7146 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7147 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7148 }
7149 } % 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.

```

7150 \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

```

7151 \AddToHook{begindocument}{
7152     \definecolor{green}{rgb}{0,.5,0}
7153     \definecolor{purple}{cmyk}{.3,1,0,.17}
7154 }

```

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`.

```

7155 % \def\STpresent#1{\textcolor{blue}{#1}}
7156 \def\defemph#1{\textcolor{magenta}{#1}}
7157 \def\symrefemph#1{\textcolor{cyan}{#1}}
7158 \def\compemph#1{\textcolor{blue}{#1}}
7159 \def\titleemph#1{\textcolor{blue}{#1}}
7160 \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.

```

7161 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7162 \def\smalltextwarning{
7163     \pgfuseimage{miko@small@dbend}
7164     \xspace
7165 }
7166 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7167 \newrobustcmd\textwarning{
7168     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7169     \xspace
7170 }
7171 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}

```

```

7172 \newrobustcmd\bigtextwarning{
7173   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7174   \xspace
7175 }
(End definition for \textwarning. This function is documented on page 55.)
7176 \newrobustcmd\putgraphicsat[3]{
7177   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7178 }
7179 \newrobustcmd\putat[2]{
7180   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7181 }

```

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`.

```

7182 \stex_html_backend:F {
7183   \bool_if:NT \c__notesslides_sectocframes_bool {
7184     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7185       \newcounter{chapter}\counterwithin*{section}{chapter}
7186     }{
7187       \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7188         \newcounter{chapter}\counterwithin*{section}{chapter}
7189       }
7190     }
7191   }
7192 }

```

`\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
7193 \def\part@prefix{}
7194 \@ifpackageloaded{document-structure}{}{
7195   \str_case:VnF \__notesslidesstopsect {
7196     {part}{
7197       \int_set:Nn \l_document_structure_section_level_int {0}
7198       \def\thesection{\arabic{chapter}.\arabic{section}}
7199       \def\part@prefix{\arabic{chapter}.}
7200     }
7201     {chapter}{
7202       \int_set:Nn \l_document_structure_section_level_int {1}
7203       \def\thesection{\arabic{chapter}.\arabic{section}}
7204       \def\part@prefix{\arabic{chapter}.}
7205     }
7206   }{
7207     \int_set:Nn \l_document_structure_section_level_int {2}
7208     \def\part@prefix{}
7209   }
7210 }
7211
7212 \bool_if:NF \c__notesslides_notes_bool { % only in slides

```

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

The new counters are used in the `sfragment` environment that choses the L^AT_EX sectioning macros according to \section@level.

`sfragment`

```

7213 \renewenvironment{sfragment}[2][]{
7214   \_document_structure_sfragment_args:n { #1 }
7215   \int_incr:N \l_document_structure_section_level_int
7216   \bool_if:NT \c_notesslides_sectocframes_bool {
7217     \stepcounter{slide}
7218     \begin{frame}[noframenumbering]
7219     \vfill\Large\centering
7220     \red{
7221       \ifcase\l_document_structure_section_level_int\or
7222         \stepcounter{part}
7223         \def\_notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7224         \def\currentsectionlevel{\omdoc@part@kw}
7225       \or
7226         \stepcounter{chapter}
7227         \def\_notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7228         \def\currentsectionlevel{\omdoc@chapter@kw}
7229       \or
7230         \stepcounter{section}
7231         \def\_notesslideslabel{\part@prefix\arabic{section}}
7232         \def\currentsectionlevel{\omdoc@section@kw}
7233       \or
7234         \stepcounter{subsection}
7235         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7236         \def\currentsectionlevel{\omdoc@subsection@kw}
7237       \or
7238         \stepcounter{subsubsection}
7239         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7240         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7241       \or
7242         \stepcounter{paragraph}
7243         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7244         \def\currentsectionlevel{\omdoc@paragraph@kw}
7245       \else
7246         \def\_notesslideslabel{}
7247         \def\currentsectionlevel{\omdoc@paragraph@kw}
7248       \fi% end ifcase
7249       \_notesslideslabel%\sref@label@id\_notesslideslabel
7250       \quad #2%
7251     }%
7252     \vfill%
7253     \end{frame}%
7254   }
7255   \str_if_empty:NF \l_document_structure_sfragment_id_str {
7256     \stex_ref_new_doc_target:n\l_document_structure_sfragment_id_str
7257   }
7258 }{}
7259 }
```

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

```

7260 \def\inserttheorembodyfont{\normalfont}
7261 %\bool_if:NF \c__notesslides_notes_bool {
7262 % \defbeamertemplate{theorem begin}{miko}
7263 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7264 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7265 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7266 % \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

7267 % \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.

```

7268 % \expandafter\def\csname Parent2\endcsname{}
7269 %}
7270
7271 \AddToHook{begindocument}{ % this does not work for some reason
7272 \setbeamertemplate{theorems}[ams style]
7273 }
7274 \bool_if:NT \c__notesslides_notes_bool {
7275 \renewenvironment{columns}[1][{}%
7276 \par\noindent%
7277 \begin{minipage}%
7278 \slidewidth\centering\leavevmode%
7279 }{}%
7280 \end{minipage}\par\noindent%
7281 }%
7282 \newsavebox\columnbox%
7283 \renewenvironment<>{column}[2][{}%
7284 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7285 }{}%
7286 \end{minipage}\end{lrbox}\usebox\columnbox%
7287 }%
7288 }
7289 \bool_if:NTF \c__notesslides_noproblems_bool {
7290 \newenvironment{problems}{}{}
7291 }{
7292 \excludacomment{problems}
7293 }
```

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.

```

7294 \gdef\printexcursions{}
7295 \newcommand\excursionref[2]{% label, text
7296 \bool_if:NT \c__notesslides_notes_bool {
7297 \begin{sparagraph}[title=Excursion]
7298 #2 \sref[fallback=the appendix]{#1}.
7299 \end{sparagraph}}
```

```

7300 }
7301 }
7302 \newcommand\activate@excursion[2][]{
7303   \gappto\printexcursions{\inputref{#1}{#2}}
7304 }
7305 \newcommand\excursion[4][]{% repos, label, path, text
7306   \bool_if:NT \c__notesslides_notes_bool {
7307     \activate@excursion{#1}{#3}\excursionref{#2}{#4}
7308   }
7309 }

```

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

\excursiongroup

```

7310 \keys_define:nn{notesslides / excursiongroup }{
7311   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7312   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7313   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7314 }
7315 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7316   \tl_clear:N \l__notesslides_excursion_intro_tl
7317   \str_clear:N \l__notesslides_excursion_id_str
7318   \str_clear:N \l__notesslides_excursion_mhrepos_str
7319   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7320 }
7321 \newcommand\excursiongroup[1][]{
7322   \__notesslides_excursion_args:n{ #1 }
7323   \ifdefempty\printexcursions{}% only if there are excursions
7324   {\begin{note}
7325     \begin{sfragment}[#1]{Excursions}%
7326     \ifdefempty\l__notesslides_excursion_intro_tl{\{
7327       \inputref[\l__notesslides_excursion_mhrepos_str]{
7328         \l__notesslides_excursion_intro_tl
7329       }
7330     }
7331     \printexcursions%
7332     \end{sfragment}
7333   }\end{note}}
7334 }
7335 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7336 \</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.

```
7337 <*package>
7338 <@@=problems>
7339 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7340 \RequirePackage{l3keys2e,stex}
7341
7342 \keys_define:nn { problem / pkg }{
7343   notes      .default:n    = { true },
7344   notes      .bool_set:N   = \c__problems_notes_bool,
7345   gnotes     .default:n    = { true },
7346   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7347   hints      .default:n    = { true },
7348   hints      .bool_set:N   = \c__problems_hints_bool,
7349   solutions  .default:n    = { true },
7350   solutions  .bool_set:N   = \c__problems_solutions_bool,
7351   pts        .default:n    = { true },
7352   pts        .bool_set:N   = \c__problems_pts_bool,
7353   min        .default:n    = { true },
7354   min        .bool_set:N   = \c__problems_min_bool,
7355   boxed      .default:n    = { true },
7356   boxed      .bool_set:N   = \c__problems_boxed_bool,
7357   unknown    .code:n       = {}
7358 }
7359 \newif\ifsolutions
7360
7361 \ProcessKeysOptions{ problem / pkg }
7362 \bool_if:NTF \c__problems_solutions_bool {
7363   \solutionstrue
7364 }{
7365   \solutionsfalse
7366 }
```

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

```
7367 \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.

```
7368 \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.

```
7369 \def\prob@problem@kw{Problem}
7370 \def\prob@solution@kw{Solution}
7371 \def\prob@hint@kw{Hint}
7372 \def\prob@note@kw{Note}
7373 \def\prob@gnote@kw{Grading}
7374 \def\prob@pt@kw{pt}
7375 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7376 \AddToHook{begindocument}{
7377   \ltx@ifpackageloaded{babel}{
7378     \makeatletter
7379     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7380     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7381       \input{problem-ngerman.ldf}
7382     }
7383     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7384       \input{problem-finnish.ldf}
7385     }
7386     \clist_if_in:NnT \l_tmpa_clist {french}{
7387       \input{problem-french.ldf}
7388     }
7389     \clist_if_in:NnT \l_tmpa_clist {russian}{
7390       \input{problem-russian.ldf}
7391     }
7392     \makeatother
7393   }{}
7394 }
```

39.2 Problems and Solutions

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

```
7395 \keys_define:nn{ problem / problem }{
7396   id      .str_set_x:N = \l__problems_prob_id_str,
7397   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7398   min     .tl_set:N    = \l__problems_prob_min_tl,
7399   title   .tl_set:N    = \l__problems_prob_title_tl,
7400   type    .tl_set:N    = \l__problems_prob_type_tl,
7401   imports .tl_set:N    = \l__problems_prob_imports_tl,
7402   name    .str_set_x:N = \l__problems_prob_name_str,
7403   refnum  .int_set:N   = \l__problems_prob_refnum_int
```



```

7404 }
7405 \cs_new_protected:Nn \__problems_prob_args:n {
7406   \str_clear:N \l__problems_prob_id_str
7407   \str_clear:N \l__problems_prob_name_str
7408   \tl_clear:N \l__problems_prob_pts_tl
7409   \tl_clear:N \l__problems_prob_min_tl
7410   \tl_clear:N \l__problems_prob_title_tl
7411   \tl_clear:N \l__problems_prob_type_tl
7412   \tl_clear:N \l__problems_prob_imports_tl
7413   \int_zero_new:N \l__problems_prob_refnum_int
7414   \keys_set:nn { problem / problem }{ #1 }
7415   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7416     \let\l__problems_prob_refnum_int\undefined
7417   }
7418 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7419 \newcounter{problem}[section]
7420 \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.

```

7421 \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

```

7422 \newcommand\prob@number{
7423   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7424     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7425   }{
7426     \int_if_exist:NTF \l__problems_prob_refnum_int {
7427       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7428     }{
7429       \prob@label\theproblem
7430     }
7431   }
7432 }

```

(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.

```

7433 \newcommand\prob@title[3]{%
7434   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7435     #2 \l__problems_inclprob_title_tl #3
7436   }{
7437     \tl_if_exist:NTF \l__problems_prob_title_tl {
7438       #2 \l__problems_prob_title_tl #3
7439     }{
7440       #1

```

```

7441     }
7442   }
7443 }

```

(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.

```

7444 \def\prob@heading{
7445   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7446   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7447 }

```

(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`

```

7448 \newenvironment{sproblem}[1][{}]{
7449   \__problems_prob_args:n{#1}%\sref@target%
7450   \@in@omtexttrue% we are in a statement (for inline definitions)
7451   \stepcounter{problem}\record@problem
7452   \def\current@section@level{\prob@problem@kw}
7453
7454   \str_if_empty:NT \l__problems_prob_name_str {
7455     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7456     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7457     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7458   }
7459
7460   \stex_if_do_html:T{
7461     \tl_if_empty:NF \l__problems_prob_title_tl {
7462       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7463     }
7464   }
7465
7466   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7467
7468   \stex_reactivate_macro:N \STEXexport
7469   \stex_reactivate_macro:N \importmodule
7470   \stex_reactivate_macro:N \symdecl
7471   \stex_reactivate_macro:N \notation
7472   \stex_reactivate_macro:N \symdef
7473
7474   \stex_if_do_html:T{
7475     \begin{stex_annotate_env} {problem} {
7476       \l_stex_module_ns_str ? \l_stex_module_name_str
7477     }
7478
7479     \stex_annotate_invisible:nnn{header}{} {
7480       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7481     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7482     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7483         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7484     }
7485 }
7486 }
7487
7488 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7489
7490
7491 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7492     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7493 }{
7494     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7495 }
7496 \str_if_exist:NTF \l__problems_inclprob_id_str {
7497     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7498 }{
7499     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7500 }
7501
7502
7503 \stex_if_smsmode:F {
7504     \clist_set:No \l_tmpa_clist \sproblemtype
7505     \tl_clear:N \l_tmpa_tl
7506     \clist_map_inline:Nn \l_tmpa_clist {
7507         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7508             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7509         }
7510     }
7511     \tl_if_empty:NTF \l_tmpa_tl {
7512         \__problems_sproblem_start:
7513     }{
7514         \l_tmpa_tl
7515     }
7516 }
7517 \stex_ref_new_doc_target:n \sproblemid
7518 \stex_smsmode_do:
7519 }{
7520     \__stex_modules_end_module:
7521     \stex_if_smsmode:F{
7522         \clist_set:No \l_tmpa_clist \sproblemtype
7523         \tl_clear:N \l_tmpa_tl
7524         \clist_map_inline:Nn \l_tmpa_clist {
7525             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7526                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7527             }
7528         }
7529         \tl_if_empty:NTF \l_tmpa_tl {
7530             \__problems_sproblem_end:
7531         }{
7532             \l_tmpa_tl
7533         }
7534     }

```

```

7535 \stex_if_do_html:T{
7536   \end{stex_annotate_env}
7537 }
7538
7539 \smallskip
7540 }
7541
7542 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7543
7544
7545
7546 \cs_new_protected:Nn \__problems_sproblem_start: {
7547   \par\noindent\textbf{\prob@heading\show@pts\show@min\\\ignorespacesandpars
7548 }
7549 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7550
7551 \newcommand\stexpatchproblem[3][] {
7552   \str_set:Nx \l_tmpa_str{ #1 }
7553   \str_if_empty:NTF \l_tmpa_str {
7554     \tl_set:Nn \__problems_sproblem_start: { #2 }
7555     \tl_set:Nn \__problems_sproblem_end: { #3 }
7556   }{
7557     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7558     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7559   }
7560 }
7561
7562
7563 \bool_if:NT \c__problems_boxed_bool {
7564   \surroundwithmdframed{problem}
7565 }

```

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

```

7566 \def\record@problem{
7567   \protected@write\@auxout{}
7568   {
7569     \string\@problem{\prob@number}
7570     {
7571       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7572         \l__problems_inclprob_pts_tl
7573       }{
7574         \l__problems_prob_pts_tl
7575       }
7576     }%
7577     {
7578       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7579         \l__problems_inclprob_min_tl
7580       }{
7581         \l__problems_prob_min_tl
7582       }
7583     }
7584   }
7585 }

```

(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).

```
7586 \def\@problem#1#2#3{}
```

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

`solution` The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```
7587 \keys_define:nn { problem / solution }{
7588   id          .str_set_x:N = \l__problems_solution_id_str ,
7589   for         .tl_set:N   = \l__problems_solution_for_tl ,
7590   height      .dim_set:N  = \l__problems_solution_height_dim ,
7591   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7592   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7593   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7594 }
7595 \cs_new_protected:Nn \__problems_solution_args:n {
7596   \str_clear:N \l__problems_solution_id_str
7597   \tl_clear:N \l__problems_solution_for_tl
7598   \tl_clear:N \l__problems_solution_srccite_tl
7599   \clist_clear:N \l__problems_solution_creators_clist
7600   \clist_clear:N \l__problems_solution_contributors_clist
7601   \dim_zero:N \l__problems_solution_height_dim
7602   \keys_set:nn { problem / solution }{ #1 }
7603 }
```

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

```
7604 \newcommand\@startsolution[1][]{
7605   \__problems_solution_args:n { #1 }
7606   \@in@omtexttrue% we are in a statement.
7607   \bool_if:NF \c__problems_boxed_bool { \hrule }
7608   \smallskip\noindent
7609   {\textbf{\prob@solution@kw :}\enspace}
7610   \begin{small}
7611   \def\current@section@level{\prob@solution@kw}
7612   \ignorespacesandpars
7613 }
```

`\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.

```
7614 \box_new:N \l__problems_solution_box
7615 \newenvironment{solution}[1][]{
7616   \stex_html_backend:TF{
7617     \stex_if_do_html:T{
7618       \begin{stex_annotate_env}{solution}{}}
7619     }
7620   }{
7621     \setbox\l__problems_solution_box\vbox\bgroup
7622     \par\smallskip\hrule\smallskip
7623     \noindent\textbf{Solution:}~
7624   }
7625 }{
7626   \stex_html_backend:TF{
```

```

7627 \stex_if_do_html:T{
7628 \end{stex_annotate_env}
7629 }
7630 }{
7631 \smallskip\hrule
7632 \egroup
7633 \bool_if:NT \c__problems_solutions_bool {
7634 \box\l__problems_solution_box
7635 }
7636 }
7637 }
7638
7639 \newcommand\startsolutions{
7640 \bool_set_true:N \c__problems_solutions_bool
7641 % \specialcomment{solution}{\@startsolution}{
7642 % \bool_if:NF \c__problems_boxed_bool {
7643 % \hrule\medskip
7644 % }
7645 % \end{small}%
7646 % }
7647 % \bool_if:NT \c__problems_boxed_bool {
7648 % \surroundwithmdframed{solution}
7649 % }
7650 }

```

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

\stopsolutions

```

7651 \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.

```

7652 \ifsolutions
7653 \startsolutions
7654 \else
7655 \stopsolutions
7656 \fi

```

exnote

```

7657 \bool_if:NTF \c__problems_notes_bool {
7658 \newenvironment{exnote}[1][{}]{
7659 \par\smallskip\hrule\smallskip
7660 \noindent\textbf{\prob@note@kw :~ }\small
7661 }{
7662 \smallskip\hrule
7663 }
7664 }{
7665 \excludecomment{exnote}
7666 }

```

hint

```

7667 \bool_if:NTF \c__problems_notes_bool {
7668 \newenvironment{hint}[1][{}]{
7669 \par\smallskip\hrule\smallskip

```

```

7670 \noindent\textbf{\prob@hint@kw :~ }\small
7671 }{
7672 \smallskip\hrule
7673 }
7674 \newenvironment{exhint}[1][]{
7675 \par\smallskip\hrule\smallskip
7676 \noindent\textbf{\prob@hint@kw :~ }\small
7677 }{
7678 \smallskip\hrule
7679 }
7680 }{
7681 \excludecomment{hint}
7682 \excludecomment{exhint}
7683 }

```

gnote

```

7684 \bool_if:NTF \c__problems_notes_bool {
7685 \newenvironment{gnote}[1][]{
7686 \par\smallskip\hrule\smallskip
7687 \noindent\textbf{\prob@gnote@kw :~ }\small
7688 }{
7689 \smallskip\hrule
7690 }
7691 }{
7692 \excludecomment{gnote}
7693 }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7694 \newenvironment{mcb}{
7695 \begin{enumerate}
7696 }{
7697 \end{enumerate}
7698 }

```

we define the keys for the mcb macro

```

7699 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7700 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7701 \bool_set_true:N #1
7702 }{
7703 \bool_set_false:N #1
7704 }
7705 }
7706 \keys_define:nn { problem / mcb }{
7707 id .str_set_x:N = \l__problems_mcc_id_str ,
7708 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7709 T .default:n = { false } ,
7710 T .bool_set:N = \l__problems_mcc_t_bool ,
7711 F .default:n = { false } ,
7712 F .bool_set:N = \l__problems_mcc_f_bool ,

```

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

```

7713 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7714 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7715 }
7716 \cs_new_protected:Nn \l__problems_mcc_args:n {
7717   \str_clear:N \l__problems_mcc_id_str
7718   \tl_clear:N \l__problems_mcc_feedback_tl
7719   \bool_set_false:N \l__problems_mcc_t_bool
7720   \bool_set_false:N \l__problems_mcc_f_bool
7721   \tl_clear:N \l__problems_mcc_Ttext_tl
7722   \tl_clear:N \l__problems_mcc_Ftext_tl
7723   \str_clear:N \l__problems_mcc_id_str
7724   \keys_set:nn { problem / mcc }{ #1 }
7725 }

```

\mcc

```

7726 \def\mccTrueText{\textbf{(true)}~}}
7727 \def\mccFalseText{\textbf{(false)}~}}
7728 \newcommand\mcc[2][]{}
7729   \l__problems_mcc_args:n{ #1 }
7730   \item[{$\Box$}] #2
7731   \ifsolutions
7732     \l
7733     \bool_if:NT \l__problems_mcc_t_bool {
7734       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7735     }
7736     \bool_if:NT \l__problems_mcc_f_bool {
7737       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7738     }
7739     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7740       \emph{(\l__problems_mcc_feedback_tl)}
7741     }
7742   \fi
7743 } %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.

```

7744
7745 \keys_define:nn{ problem / inclproblem }{
7746   id      .str_set_x:N      = \l__problems_inclprob_id_str,
7747   pts     .tl_set:N         = \l__problems_inclprob_pts_tl,
7748   min     .tl_set:N         = \l__problems_inclprob_min_tl,
7749   title   .tl_set:N         = \l__problems_inclprob_title_tl,
7750   refnum  .int_set:N        = \l__problems_inclprob_refnum_int,
7751   type    .tl_set:N         = \l__problems_inclprob_type_tl,
7752   mhrepos .str_set_x:N      = \l__problems_inclprob_mhrepos_str
7753 }
7754 \cs_new_protected:Nn \__problems_inclprob_args:n {
7755   \str_clear:N \l__problems_prob_id_str

```



```

7756 \tl_clear:N \l__problems_inclprob_pts_tl
7757 \tl_clear:N \l__problems_inclprob_min_tl
7758 \tl_clear:N \l__problems_inclprob_title_tl
7759 \tl_clear:N \l__problems_inclprob_type_tl
7760 \int_zero_new:N \l__problems_inclprob_refnum_int
7761 \str_clear:N \l__problems_inclprob_mhrepos_str
7762 \keys_set:nn { problem / inclproblem }{ #1 }
7763 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7764   \let\l__problems_inclprob_pts_tl\undefined
7765 }
7766 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7767   \let\l__problems_inclprob_min_tl\undefined
7768 }
7769 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7770   \let\l__problems_inclprob_title_tl\undefined
7771 }
7772 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7773   \let\l__problems_inclprob_type_tl\undefined
7774 }
7775 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7776   \let\l__problems_inclprob_refnum_int\undefined
7777 }
7778 }
7779
7780 \cs_new_protected:Nn \__problems_inclprob_clear: {
7781   \let\l__problems_inclprob_id_str\undefined
7782   \let\l__problems_inclprob_pts_tl\undefined
7783   \let\l__problems_inclprob_min_tl\undefined
7784   \let\l__problems_inclprob_title_tl\undefined
7785   \let\l__problems_inclprob_type_tl\undefined
7786   \let\l__problems_inclprob_refnum_int\undefined
7787   \let\l__problems_inclprob_mhrepos_str\undefined
7788 }
7789 \__problems_inclprob_clear:
7790
7791 \newcommand\includeproblem[2][ ]{
7792   \__problems_inclprob_args:n{ #1 }
7793   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7794     \stex_html_backend:TF {
7795       \str_clear:N \l_tmpa_str
7796       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7797         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7798       }
7799       \stex_annotate_invisible:nnn{includeproblem}{
7800         \l_tmpa_str / #2
7801       }{}
7802     }{
7803       \begingroup
7804         \inputreftrue
7805         \tl_if_empty:nTF{ ##1 }{
7806           \input{#2}
7807         }{
7808           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7809         }

```

```

7810     \endgroup
7811   }
7812 }
7813 \__problems_inclprob_clear:
7814 }

```

(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.

```

7815 \AddToHook{enddocument}{
7816   \bool_if:NT \c__problems_pts_bool {
7817     \message{Total:~\arabic{pts}~points}
7818   }
7819   \bool_if:NT \c__problems_min_bool {
7820     \message{Total:~\arabic{min}~minutes}
7821   }
7822 }

```

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

```

7823 \def\pts#1{
7824   \bool_if:NT \c__problems_pts_bool {
7825     \marginpar{#1~\prob@pt@kw}
7826   }
7827 }
7828 \def\min#1{
7829   \bool_if:NT \c__problems_min_bool {
7830     \marginpar{#1~\prob@min@kw}
7831   }
7832 }

```

`\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.

```

7833 \newcounter{pts}
7834 \def\show@pts{
7835   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7836     \bool_if:NT \c__problems_pts_bool {
7837       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7838       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7839     }
7840   }{
7841     \tl_if_exist:NT \l__problems_prob_pts_tl {
7842       \bool_if:NT \c__problems_pts_bool {
7843         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7844           \tl_set:Nn \l__problems_prob_pts_tl {0}
7845         }
7846         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7847         \addtocounter{pts}{\l__problems_prob_pts_tl}
7848       }
7849     }

```

```

7850 }
7851 }

```

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

\show@min

```

7852 \newcounter{min}
7853 \def\show@min{
7854   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7855     \bool_if:NT \c__problems_min_bool {
7856       \marginpar{\l__problems_inclprob_pts_tl\ min}
7857       \addtocounter{min}{\l__problems_inclprob_min_tl}
7858     }
7859   }{
7860     \tl_if_exist:NT \l__problems_prob_min_tl {
7861       \bool_if:NT \c__problems_min_bool {
7862         \tl_if_empty:NT\l__problems_prob_min_tl{
7863           \tl_set:Nn \l__problems_prob_min_tl {0}
7864         }
7865         \marginpar{\l__problems_prob_min_tl\ min}
7866         \addtocounter{min}{\l__problems_prob_min_tl}
7867       }
7868     }
7869   }
7870 }
7871 \</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.

```
7872 \*package>
7873 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7874 \RequirePackage{13keys2e}
7875
7876 \newif\iftest\testfalse
7877 \DeclareOption{test}{\testtrue}
7878 \newif\ifmultiple\multiplefalse
7879 \DeclareOption{multiple}{\multipletrue}
7880 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7881 \ProcessOptions
```

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

```
7882 \RequirePackage{keyval}[1997/11/10]
7883 \RequirePackage{problem}
```

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

```
7884 \newcommand\hwexam@assignment@kw{Assignment}
7885 \newcommand\hwexam@given@kw{Given}
7886 \newcommand\hwexam@due@kw{Due}
7887 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7888 blank~for~extra~space}
7889 \def\hwexam@minutes@kw{minutes}
7890 \newcommand\correction@probs@kw{prob.}
7891 \newcommand\correction@pts@kw{total}
7892 \newcommand\correction@reached@kw{reached}
7893 \newcommand\correction@sum@kw{Sum}
7894 \newcommand\correction@grade@kw{grade}
7895 \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

```

7896 \AddToHook{begindocument}{
7897 \ltx@ifpackageloaded{babel}{
7898 \makeatletter
7899 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7900 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7901 \input{hwexam-ngerman.ldf}
7902 }
7903 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7904 \input{hwexam-finnish.ldf}
7905 }
7906 \clist_if_in:NnT \l_tmpa_clist {french}{
7907 \input{hwexam-french.ldf}
7908 }
7909 \clist_if_in:NnT \l_tmpa_clist {russian}{
7910 \input{hwexam-russian.ldf}
7911 }
7912 \makeatother
7913 }{}
7914 }
7915

```

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.

```

7916 \newcounter{assignment}
7917 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7918 \keys_define:nn { hwexam / assignment } {
7919 id .str_set:N = \l_@@_assign_id_str,
7920 number .int_set:N = \l_@@_assign_number_int,
7921 title .tl_set:N = \l_@@_assign_title_tl,
7922 type .tl_set:N = \l_@@_assign_type_tl,
7923 given .tl_set:N = \l_@@_assign_given_tl,
7924 due .tl_set:N = \l_@@_assign_due_tl,
7925 loadmodules .code:n = {
7926 \bool_set_true:N \l_@@_assign_loadmodules_bool
7927 }
7928 }
7929 \cs_new_protected:Nn \_@@_assignment_args:n {
7930 \str_clear:N \l_@@_assign_id_str
7931 \int_set:Nn \l_@@_assign_number_int {-1}
7932 \tl_clear:N \l_@@_assign_title_tl
7933 \tl_clear:N \l_@@_assign_type_tl
7934 \tl_clear:N \l_@@_assign_given_tl
7935 \tl_clear:N \l_@@_assign_due_tl
7936 \bool_set_false:N \l_@@_assign_loadmodules_bool
7937 \keys_set:nn { hwexam / assignment }{ #1 }
7938 }

```

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.

```

7939 \newcommand\given@due[2]{
7940 \bool_lazy_all:nF {
7941 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7942 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7943 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7944 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7945 }{ #1 }
7946
7947 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7948 \tl_if_empty:NF \l_@@_assign_given_tl {
7949 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7950 }
7951 }{
7952 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7953 }
7954
7955 \bool_lazy_or:nnF {
7956 \bool_lazy_and_p:nn {
7957 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7958 }{
7959 \tl_if_empty_p:V \l_@@_assign_due_tl
7960 }
7961 }{
7962 \bool_lazy_and_p:nn {
7963 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7964 }{
7965 \tl_if_empty_p:V \l_@@_assign_due_tl
7966 }
7967 }{ ,~ }
7968
7969 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7970 \tl_if_empty:NF \l_@@_assign_due_tl {
7971 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7972 }
7973 }{
7974 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7975 }
7976
7977 \bool_lazy_all:nF {
7978 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7979 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7980 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7981 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7982 }{ #2 }
7983 }

```

`\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.

```

7984 \newcommand\assignment@title[3]{
7985 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7986 \tl_if_empty:NTF \l_@@_assign_title_tl {
7987 #1
7988 }{
7989 #2\l_@@_assign_title_tl#3
7990 }
7991 }{
7992 #2\l_@@_inclasssign_title_tl#3
7993 }
7994 }

```

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

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

```

7995 \newcommand\assignment@number{
7996 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
7997 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7998 \arabic{assignment}
7999 } {
8000 \int_use:N \l_@@_assign_number_int
8001 }
8002 }{
8003 \int_use:N \l_@@_inclasssign_number_int
8004 }
8005 }

```

(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.

```

8006 \newenvironment{assignment}[1][]{
8007 \_@@_assignment_args:n { #1 }
8008 %\sref@target
8009 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8010 \global\stepcounter{assignment}
8011 }{
8012 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8013 }
8014 \setcounter{problem}{0}
8015 \renewcommand\prob@label[1]{\assignment@number.##1}
8016 \def\current@section@level{\document@hwexamtype}
8017 %\sref@label{id}{\document@hwexamtype \thesection}
8018 \begin{@assignment}
8019 }{
8020 \end{@assignment}
8021 }

```

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

```

8022 \def\ass@title{
8023 {\protect\document@hwexamtype}\arabic{assignment}
8024 \assignment@title{}\{;\}\{;\} -- \given@due{}\{;\}
8025 }
8026 \ifmultiple
8027 \newenvironment{@assignment}{
8028 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8029 \begin{sfragment}[loadmodules]{\ass@title}
8030 }{
8031 \begin{sfragment}{\ass@title}
8032 }
8033 }{
8034 \end{sfragment}
8035 }

```

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

```

8036 \else
8037 \newenvironment{@assignment}{
8038 \begin{center}\bf
8039 \Large@title\strut\
8040 \document@hwexamtype}\arabic{assignment}\assignment@title{}\{;\}\{;\}\{;\}
8041 \large\given@due{--;\}\{;\}--}
8042 \end{center}
8043 }{}
8044 \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.

```

8045 \keys_define:nn { hwexam / inclassignment } {
8046 %id .str_set_x:N = \l_@@_assign_id_str,
8047 number .int_set:N = \l_@@_inclassign_number_int,
8048 title .tl_set:N = \l_@@_inclassign_title_tl,
8049 type .tl_set:N = \l_@@_inclassign_type_tl,
8050 given .tl_set:N = \l_@@_inclassign_given_tl,
8051 due .tl_set:N = \l_@@_inclassign_due_tl,
8052 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8053 }
8054 \cs_new_protected:Nn \_@@_inclassignment_args:n {
8055 \int_set:Nn \l_@@_inclassign_number_int {-1}
8056 \tl_clear:N \l_@@_inclassign_title_tl
8057 \tl_clear:N \l_@@_inclassign_type_tl
8058 \tl_clear:N \l_@@_inclassign_given_tl
8059 \tl_clear:N \l_@@_inclassign_due_tl
8060 \str_clear:N \l_@@_inclassign_mhrepos_str
8061 \keys_set:nn { hwexam / inclassignment }{ #1 }
8062 }
8063 \_@@_inclassignment_args:n {}
8064
8065 \newcommand\inputassignment[2][{}]{

```



```

8066 \_@@_inclassignment_args:n { #1 }
8067 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
8068 \input{#2}
8069 }{
8070 \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
8071 \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
8072 }
8073 }
8074 \_@@_inclassignment_args:n {}
8075 }
8076 \newcommand\includeassignment[2][ ]{
8077 \newpage
8078 \inputassignment[#1]{#2}
8079 }

```

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

40.4 Typesetting Exams

\quizheading

```

8080 \ExplSyntaxOff
8081 \newcommand\quizheading[1]{%
8082 \def\@tas{#1}%
8083 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8084 \ifx\@tas\@empty\else%
8085 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8086 \fi%
8087 }
8088 \ExplSyntaxOn

```

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

\testheading

```

8089
8090 \def\hwexamheader{\input{hwexam-default.header}}
8091
8092 \def\hwexamminutes{
8093 \tl_if_empty:NTF \testheading@duration {
8094 {\testheading@min}~\hwexam@minutes@kw
8095 }{
8096 \testheading@duration
8097 }
8098 }
8099
8100 \keys_define:nn { hwexam / testheading } {
8101 min .tl_set:N = \testheading@min,
8102 duration .tl_set:N = \testheading@duration,
8103 reqpts .tl_set:N = \testheading@reqpts,
8104 tools .tl_set:N = \testheading@tools
8105 }
8106 \cs_new_protected:Nn \_@@_testheading_args:n {
8107 \tl_clear:N \testheading@min
8108 \tl_clear:N \testheading@duration

```

```

8109 \tl_clear:N \testheading@reqpts
8110 \tl_clear:N \testheading@tools
8111 \keys_set:nn { hwexam / testheading }{ #1 }
8112 }
8113 \newenvironment{testheading}[1][ ]{
8114 \_@@_testheading_args:n{ #1 }
8115 \newcount\check@time\check@time=\testheading@min
8116 \advance\check@time by -\theassignment@totalmin
8117 \newif\if@bonuspoints
8118 \tl_if_empty:NTF \testheading@reqpts {
8119 \@bonuspointsfalse
8120 }{
8121 \newcount\bonus@pts
8122 \bonus@pts=\theassignment@totalpts
8123 \advance\bonus@pts by -\testheading@reqpts
8124 \edef\bonus@pts{\the\bonus@pts}
8125 \@bonuspointstrue
8126 }
8127 \edef\check@time{\the\check@time}
8128
8129 \makeatletter\hwexamheader\makeatother
8130 }{
8131 \newpage
8132 }

```

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

\testspace

```

8133 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8134 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8135 \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.

```

8136 <@=problems>
8137 \renewcommand\@problem[3]{
8138 \stepcounter{assignment@probs}
8139 \def\__problemspts{#2}
8140 \ifx\__problemspts\@empty\else
8141 \addtocounter{assignment@totalpts}{#2}
8142 \fi
8143 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8144 \xdef\correction@probs{\correction@probs & #1}%
8145 \xdef\correction@pts{\correction@pts & #2}
8146 \xdef\correction@reached{\correction@reached &}

```

```

8147 }
8148 <@@=hwexam>

```

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

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

```

8149 \newcounter{assignment@probs}
8150 \newcounter{assignment@totalpts}
8151 \newcounter{assignment@totalmin}
8152 \def\correction@probs{\correction@probs@kw}
8153 \def\correction@pts{\correction@pts@kw}
8154 \def\correction@reached{\correction@reached@kw}
8155 \stepcounter{assignment@probs}
8156 \newcommand\correction@table{
8157 \resizebox{\textwidth}{!}{%
8158 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8159 &\multicolumn{\theassignment@probs}{c|}|%|
8160 {\footnotesize\correction@forgrading@kw} &\\ \hline
8161 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8162 \correction@pts & \theassignment@totalpts & \\ \hline
8163 \correction@reached & & \[.7cm]\hline
8164 \end{tabular}}
8165 \end{package}

```

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

40.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\bierfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

\newcommand\bierglas{{\bierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\bierglas}

```

Chapter 41

References

- [Bus+04] Stephen Buswell et al. *The Open Math Standard, Version 2.0*. Tech. rep. The OpenMath Society, 2004. URL: <http://www.openmath.org/standard/om20>.
- [CR99] David Carlisle and Sebastian Rathz. *The graphicx package*. Part of the T_EX distribution. The Comprehensive T_EX Archive Network. 1999. URL: <https://www.tug.org/texlive/devsrc/Master/texmf-dist/doc/latex/graphics/graphicx.pdf>.
- [DCM03] The DCMI Usage Board. *DCMI Metadata Terms*. DCMI Recommendation. Dublin Core Metadata Initiative, 2003. URL: <http://dublincore.org/documents/dcmi-terms/>.
- [Koh06] Michael Kohlhase. *OMDoc – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [LMH] *LMH Scripts*. URL: <https://github.com/sLaTeX/lmhtools>.
- [MMT] *MMT – Language and System for the Uniform Representation of Knowledge*. Project web site. URL: <https://uniformal.github.io/> (visited on 01/15/2019).
- [MRK18] Dennis Müller, Florian Rabe, and Michael Kohlhase. “Theories as Types”. In: *9th International Joint Conference on Automated Reasoning*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Springer Verlag, 2018. URL: <https://kwarc.info/kohlhase/papers/ijcar18-records.pdf>.
- [Rab15] Florian Rabe. “The Future of Logic: Foundation-Independence”. In: *Logica Universalis* 10.1 (2015). 10.1007/s11787-015-0132-x; Winner of the Contest “The Future of Logic” at the World Congress on Universal Logic, pp. 1–20.
- [RK13] Florian Rabe and Michael Kohlhase. “A Scalable Module System”. In: *Information & Computation* 0.230 (2013), pp. 1–54. URL: <https://kwarc.info/frabe/Research/mmt.pdf>.
- [RT] *sLaTeX/RusTeX*. URL: <https://github.com/sLaTeX/RusTeX> (visited on 04/22/2022).

²²EDNOTE: we need an un-numbered version sfragment*

- [SIa] *sLaTeX/sTeX-IDE*. URL: <https://github.com/slatex/sTeX-IDE> (visited on 04/22/2022).
- [SIb] *sLaTeX/stexls-vscode-plugin*. URL: <https://github.com/slatex/stexls-vscode-plugin> (visited on 04/22/2022).
- [SLS] *sLaTeX/stexls*. URL: <https://github.com/slatex/stexls> (visited on 04/22/2022).
- [ST] *sTeX – An Infrastructure for Semantic Preloading of LaTeX Documents*. URL: <https://ctan.org/pkg/stex> (visited on 04/22/2022).
- [sTeX] *sTeX: A semantic Extension of TeX/LaTeX*. URL: <https://github.com/sLaTeX/sTeX> (visited on 05/11/2020).
- [Tana] Till Tantau. *beamer – A LaTeX class for producing presentations and slides*. URL: <http://ctan.org/pkg/beamer> (visited on 01/07/2014).
- [Tanb] Till Tantau. *User Guide to the Beamer Class*. URL: <http://ctan.org/macros/latex/contrib/beamer/doc/beameruserguide.pdf>.
- [TL] *TeX Live*. URL: <http://www.tug.org/texlive/> (visited on 12/11/2012).