

The sTeX3 Package Collection *

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

2022-05-22

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 3.1 (last revised 2022-05-22)

Contents

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

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

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

28	STEX-Module Inheritance Implementation	134
28.1	SMS Mode	134
28.2	Inheritance	138
29	STEX-Symbols Implementation	144
29.1	Symbol Declarations	144
29.2	Notations	152
29.3	Variables	160
30	STEX-Terms Implementation	168
30.1	Symbol Invocations	168
30.2	Terms	175
30.3	Notation Components	179
30.4	Variables	181
30.5	Sequences	184
31	STEX-Structural Features Implementation	185
31.1	Imports with modification	186
31.2	The feature environment	194
31.3	Structure	194
32	STEX-Statements Implementation	205
32.1	Definitions	205
32.2	Assertions	211
32.3	Examples	214
32.4	Logical Paragraphs	217
33	The Implementation	222
33.1	Proofs	222
33.2	Justifications	233
34	STEX-Others Implementation	234
35	STEX-Metattheory Implementation	236
36	Tikzinput Implementation	239
37	document-structure.sty Implementation	242
37.1	Package Options	242
37.2	Document Structure	243
37.3	Front and Backmatter	247
37.4	Global Variables	249
38	NotesSlides – Implementation	250
38.1	Class and Package Options	250
38.2	Notes and Slides	252
38.3	Header and Footer Lines	256
38.4	Frame Images	258
38.5	Colors and Highlighting	259
38.6	Sectioning	260
38.7	Excursions	263

39 The Implementation	265
39.1 Package Options	265
39.2 Problems and Solutions	266
39.3 Multiple Choice Blocks	273
39.4 Including Problems	274
39.5 Reporting Metadata	276
40 Implementation: The hwexam Package	278
40.1 Package Options	278
40.2 Assignments	279
40.3 Including Assignments	282
40.4 Typesetting Exams	283
40.5 Leftovers	285
41 References	286

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 sTeX : as a

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

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

2.1.2 GIT-based Setup for the sTeX Development Version

If you want use the latest and greatest sTeX packages, you can that have not even been released to CTAN, then you can directly clone them from the 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 sTeX directory. Then update your `TEXINPUTS` environment variable, e.g. by placing the following line in your `.bashrc`:

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

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

2.1.3 sTeX Archives (Manual Setup)

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

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

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

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

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

2.1.4 The sTeX IDE

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

2.1.5 Manual Setup for Active Documents and Knowledge Management Services

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

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

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

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

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

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

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

2.2 A First \LaTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

`\symname`

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

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

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

`\symref`

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

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

`\definame`
`\definiendum`

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

```

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

```

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

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

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

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

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

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

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

Remark 2.2.2:

Note that the **html** when opened in a browser will look slightly different than the **pdf** when it comes to highlighting semantic content – that is because naturally **html** allows for much more powerful features than **pdf** does. Consequently, the **html** is intended to be served by a system like MMT, which can pick up on the semantic information and offer much more powerful highlighting, linking and similar features, and being customizable by *readers* rather than being prescribed by an author.

Additionally, not all browsers (most notably Chrome) support MATHML natively, and might require additional external JavaScript libraries such as MathJax to render mathematical formulas properly.

Chapter 3

Creating sTeX Content

We can use sTeX by simply including the package with `\usepackage{stex}`, or – primarily for individual fragments to be included in other documents – by using the sTeX document class with `\documentclass{stex}` which combines the `standalone` document class with the `stex` package.

Both the `stex` package and document class offer the following options:

lang (*<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 \TeX will look for files included via `\libinput`.

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

We recommend the following additional directory structure in the `source`-folder of an \TeX archive:

- `/source/mod/` – individual \TeX modules, containing symbol declarations, notations, and `\begin{paragraph}``[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 \TeX (and associated software) of various properties of an archive. For example, the `MANIFEST.MF` of the `smglom/calculus`-archive looks like this:

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

Many of these are in fact ignored by \TeX , but some are important:

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

`source-base` or

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

```

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

```

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

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

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

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

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

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

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

Output:

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

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

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

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

$\backslash\text{setnotation}$

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

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

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

```

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

Output:

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

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

Mode-a Arguments

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

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

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

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

Example 10

Input:

```

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

```

Output:

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

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

Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

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

Mode-B Arguments

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

Example 12

Input:

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

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

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

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

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

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

In the example above:



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

3.3.6 Variables

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

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

`\svar`

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

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

`\vardef`

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

Example 19

Input:

```

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

```

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

Example 20

Input:


```

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

```

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$a_1 + \dots + a_n$

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

Example 22

Input:

```

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

```

Output:

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

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

Example 23

Input:

```

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

```

Output:

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

3.4 Module Inheritance and Structures

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

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

3.4.1 Multilinguality and Translations

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

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

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

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

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

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

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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



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



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

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

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

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

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

`\STEXexport`

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



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

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



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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

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

Output:

A `monoid` is...

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

Example 25

Input:

```

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

```

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

\instantiate

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

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

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

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

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

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

`\varinstantiate`

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

This allows us to do things like:

Example 27

Input:

```

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

```

Output:

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

.

and

Example 28

Input:

```

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

```

Output:

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

.

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 29

Input:

```

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

```

Output:

.

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

Example 30

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 The `interpretmodule` Environment

TODO: explain

Example 31

Input:

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

Output:

3.5 Primitive Symbols (The `STEX` Metatheory)

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

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

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

We make this theory part of the `STEX` collection rather than encoding it in `STEX` itself⁴

⁴EdNOTE: MK: why? continue

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

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

Example 32

Input:

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

Output:

A natural number is...

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

`\Symname`

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

Example 33

Input:

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

Output:

```
Natural numbers are...
```



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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 34

Input:

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

Output:

```
The sum of  $n$  and  $m$  is...
```

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

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



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

\arg

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

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

Example 35

Input:

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

Output:

```
Addition is...
```

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

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

Example 36

Input:

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

Output:

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

adding k yields...

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

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

Example 37

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

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

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 38

Input:

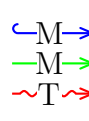

```
1 \begin{sexample}[
2   id=additionandmultiplication.ex,
3   for={addition,multiplication},
4   type={trivial,boring},
5   title={An Example}
6 ]
7   $\addition{2,3}$ is $5$, $\multiplication{2,3}$ is $6$.
8 \end{sexample}
```

Output:

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

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

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

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

`\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**

<p>1. For the induction we have to consider the following cases:</p> <p>1.1. $n = 1$: then we compute $1 = 1^2$ □</p> <p>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 $1 + 3 = 2^2 = 4$ □</p> <p>1.3. $n > 1$:</p> <p>1.3.1. Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.</p> <p>1.3.2. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.</p> <p>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.</p> <p>1.3.4. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.</p> <p>1.3.5. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □</p> <p>1.4. We have considered all the cases, so we have proven the assertion. □</p>
--

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

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

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

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

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

7.2 Modular Document Structuring

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

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

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

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

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

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

```

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

```

¹⁰EdNOTE: MK: still?

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

`\prematurestop`
`\afterprematurestop`

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

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

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

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

7.3 Slides and Course Notes

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

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

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

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



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

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

```

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

```

`\ifnotes`

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

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



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



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

`\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

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

`\setlicensing`

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

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

`\frameimage`
`\mhframeimage`

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

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

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

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

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

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

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

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

`\textwarning`

The `\textwarning` macro generates a warning sign: 

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

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

`\excursion`

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

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

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

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

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

`\excursiongroup`

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

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



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

7.4 Representing Problems and Solutions

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

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

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

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

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

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

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

Example 40

Input:

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

`\ifsolutions`

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

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

`\mcc`

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

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

If we start the solutions, then we get

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

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

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

¹²EdNOTE: MK: that did not work!

Example 42

Input:

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

Output:

Problem 7.4.3 (Functions)

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

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

\includeproblem

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

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

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

7.5 Homeworks, Quizzes and Exams

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

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

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

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-05-22

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 `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

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

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

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

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

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

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

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

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\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-</code> mode, 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 `\stex_invoke_module:n`

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

`\stex_activate_module:n`

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

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

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

12.1.2 Imports and Inheritance

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

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

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

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

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

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

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

<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>
--------------------------	--

<code>\symref</code>	<code>\symref{<symbol>}{<text>}</code> shortcut for <code>\STEXsymbol{<symbol>}! [<text>]</code>
----------------------	---

<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 `!`, it will invoke the symbol *itself* rather than its application (and continue to `\stex_term_custom:nn`), i.e. it allows to refer to `\plus![addition]` as an operation, rather than `\plus[addition of]{some}{terms}`.

<code>\STEXInternalTermMathOMSiiii</code> <code>\STEXInternalTermMathOMAiiai</code> <code>\STEXInternalTermMathOMBiiii</code>	<code><URI><fragment><precedence><body></code>
---	--

Annotates `<body>` as an OMDOC-term (OMID, OMA or OMBIND, respectively) with head symbol `<URI>`, generated by the specific notation `<fragment>` with (upwards) operator precedence `<precedence>`. Inserts parentheses according to the current downwards precedence and operator precedence.

<code>\STEXInternalTermMathArgiii</code>	<code>\stex_term_arg:nnn<int><prec><body></code>
--	--

Annotates `<body>` as the `<int>`th argument of the current OMA or OMBIND, with (downwards) argument precedence `<prec>`.

<hr/> <hr/>	<code>\STEXInternalTermMathAssocArgiiii</code>	<code>\stex_term_arg:nnn⟨int⟩⟨prec⟩⟨notation⟩⟨body⟩</code>
		Annotates $\langle body \rangle$ as the $\langle int \rangle$ th (associative) <i>sequence</i> argument (as comma-separated list of terms) of the current OMA or OMBIND, with (downwards) argument precedence $\langle prec \rangle$ and associative notation $\langle notation \rangle$.
<hr/> <hr/>	<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \TeX brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \TeX for automated bracketing (by default (and)) to $\langle left \rangle$ and $\langle right \rangle$. Note that $\langle left \rangle$ and $\langle right \rangle$ need to be allowed after <code>\left</code> and <code>\right</code> in display-mode.
<hr/> <hr/>	<code>\stex_term_custom:nn</code>	<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/> <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>	<code>\comp{⟨args⟩}</code> Marks $\langle args \rangle$ 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/> <hr/>	<code>\STEXinvisible</code>	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/> <hr/>	<code>\ellipses</code>	TODO

Chapter 15

ST_EX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

Chapter 18

sTeX-Metatheory

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput: Treating TIKZ code as images

19.1 Macros and Environments

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

Chapter 20

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

Chapter 21

NotesSlides – Slides and Course Notes

Chapter 22

`problem.sty`: An Infrastructure for formatting Problems

Chapter 23

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

Part IV
Implementation

Chapter 24

\TeX -Basics Implementation

24.1 The \TeX Document Class

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

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

```

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

```

24.2 Preliminaries

```

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

```

Package options:

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

```

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

```

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

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

116 `<@=stex_annotate>`

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

117 `\tl_new:N \l_stex_html_arg_tl`

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

`_stex_html_checkempty:n`

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

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

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

`\stex_if_do_html:TF`

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

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

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

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

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

`\stex_annotate:bnx`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

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

```

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

24.5 Babel Languages

```

157 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

```

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

```

24.6 Persistence

```

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

```

```

238 \def \stex_persist:n #1 {}
239 \def \stex_persist:x #1 {}
240 }{
241 \bool_if:NTF \c_stex_persist_write_mode_bool {
242 \iow_new:N \c__stex_persist_iow
243 \iow_open:Nn \c__stex_persist_iow{\jobname.sms}
244 \AtEndDocument{
245 \iow_close:N \c__stex_persist_iow
246 }
247 \cs_new_protected:Nn \stex_persist:n {
248 \tl_set:Nn \l_tmpa_tl { #1 }
249 \regex_replace_all:nnN { \cP\# } { \cO\# } \l_tmpa_tl
250 \regex_replace_all:nnN { \ } { \~ } \l_tmpa_tl
251 \exp_args:NNo \iow_now:Nn \c__stex_persist_iow \l_tmpa_tl
252 }
253 \cs_generate_variant:Nn \stex_persist:n {x}
254 }{
255 \def \stex_persist:n #1 {}
256 \def \stex_persist:x #1 {}
257 }
258 }

```

24.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

259 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
260 \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
261 \def#1{
262 \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
263 }
264 }

```

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

\stex_reactivate_macro:N

```

265 \cs_new_protected:Nn \stex_reactivate_macro:N {
266 \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
267 }

```

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

\ignorespacesandpars

```

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

```



```

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

```

```
334 \cs_generate_variant:Nn \stex_copy_control_sequence_ii:NNN {ccN}
```

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

\MMTrule

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

```

384     \if@minipage\else\if@nobreak\else\par\fi\fi
385   }
386 }
387
388 \end{package}

```

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

Chapter 25

STEX -MathHub Implementation

```
389 <*package>
390
391 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
392
393 <@@=stex_path>
394
395 Warnings and error messages
396 \msg_new:nnn{stex}{error/norepository}{
397   No~archive~#1~found~in~#2
398 }
399 \msg_new:nnn{stex}{error/notinarchive}{
400   Not~currently~in~an~archive,~but~\detokenize{#1}~
401   needs~one!
402 }
403 \msg_new:nnn{stex}{error/nofile}{
404   \detokenize{#1}~could~not~find~file~#2
405 }
406 \msg_new:nnn{stex}{error/twofiles}{
407   \detokenize{#1}~found~two~candidates~for~#2
408 }
```

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`

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

```

415     \seq_map_inline:Nn #1 {
416       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
417       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
418     }
419     \seq_set_eq:NN #1 \l_tmpa_tl
420   }
421   \stex_path_canonicalize:N #1
422 }
423 }
424

```

(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`

```

425 \cs_new_protected:Nn \stex_path_to_string:NN {
426   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
427 }
428
429 \cs_new:Nn \stex_path_to_string:N {
430   \seq_use:Nn #1 /
431 }

```

(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`

```

432 \str_const:Nn \c__stex_path_dot_str {.}
433 \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.

```

434 \cs_new_protected:Nn \stex_path_canonicalize:N {
435   \seq_if_empty:NF #1 {
436     \seq_clear:N \l_tmpa_seq
437     \seq_get_left:NN #1 \l_tmpa_tl
438     \str_if_empty:NT \l_tmpa_tl {
439       \seq_put_right:Nn \l_tmpa_seq {}
440     }
441     \seq_map_inline:Nn #1 {
442       \str_set:Nn \l_tmpa_tl { ##1 }
443       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
444         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
445           \seq_if_empty:NNTF \l_tmpa_seq {
446             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
447               \c__stex_path_up_str
448             }
449           }{
450             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
451             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
452               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
453                 \c__stex_path_up_str
454               }
455             }{

```

```

456         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
457     }
458 }
459 }{
460     \str_if_empty:NF \l_tmpa_tl {
461         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
462     }
463 }
464 }
465 }
466 \seq_gset_eq:NN #1 \l_tmpa_seq
467 }
468 }

```

(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}`

```

469 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
470     \seq_if_empty:NTF #1 {
471         \prg_return_false:
472     }{
473         \seq_get_left:NN #1 \l_tmpa_tl
474         \sys_if_platform_windows:TF{
475             \str_if_in:NnTF \l_tmpa_tl {:}{
476                 \prg_return_true:
477             }{
478                 \prg_return_false:
479             }
480         }{
481             \str_if_empty:NTF \l_tmpa_tl {
482                 \prg_return_true:
483             }{
484                 \prg_return_false:
485             }
486         }
487     }
488 }

```

(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`

```

489 \str_new:N\l_stex_kpsewhich_return_str
490 \cs_new_protected:Nn \stex_kpsewhich:n {\begingroup
491     \catcode'\ =12
492     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
493     \tl_gset_eq:NN \l_tmpa_tl \l_tmpa_tl
494     \endgroup
495     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
496     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
497 }

```

(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`

```

498 \sys_if_platform_windows:TF{
499   \begingroup\escapechar=-1\catcode'\=12
500   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
501   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
502   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
503   }}{
504   \stex_kpsewhich:n{-var-value~PWD}
505   }
506
507 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
508 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
509 \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

510 `<@@=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

```

511 \seq_gclear_new:N\g_stex_files_stack

```

(End definition for `\g_stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

512 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
513 \stex_path_from_string:Nn \c_stex_mainfile_seq
514   \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`

```

515 \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`

```

516 \cs_new_protected:Nn \stex_filestack_push:n {
517   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
518   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
519     \stex_path_from_string:Nn\g_stex_currentfile_seq{
520       \c_stex_pwd_str/#1
521     }

```

```

522 }
523 \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
524 \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
525 }

```

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

`\stex_filestack_pop:`

```

526 \cs_new_protected:Nn \stex_filestack_pop: {
527   \seq_if_empty:NF\g__stex_files_stack{
528     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
529   }
530   \seq_if_empty:NTF\g__stex_files_stack{
531     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
532   }{
533     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
534     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
535   }
536 }

```

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

Hooks for the current file:

```

537 \AddToHook{file/before}{
538   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
539 }
540 \AddToHook{file/after}{
541   \stex_filestack_pop:
542 }

```

25.4 MathHub Repositories

```

543 <@=stex_mathhub>

```

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

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

```

544 \str_if_empty:NTF\mathhub{
545   \sys_if_platform_windows:TF{
546     \begingroup\escapechar=-1\catcode'\=12
547     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
548     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
549     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
550   }{
551     \stex_kpsewhich:n{-var-value-MATHHUB}
552   }
553   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
554 }
555 \str_if_empty:NT \c_stex_mathhub_str {
556   \sys_if_platform_windows:TF{
557     \begingroup\escapechar=-1\catcode'\=12
558     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
559     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
560     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
561   }{

```



```

562     \stex_kpsewhich:n{-var-value~HOME}
563   }
564   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
565     \begingroup\escapechar=-1\catcode'\=12
566     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
567     \sys_if_platform_windows:T{
568       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
569     }
570     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
571     \endgroup
572     \ior_close:N \l_tmpa_ior
573   }
574 }
575 \str_if_empty:NTF\c_stex_mathhub_str{
576   \msg_warning:nn{stex}{warning/nomathhub}
577 }{
578   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
579   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
580 }
581 }{
582   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
583   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
584     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
585       \c_stex_pwd_str/\mathhub
586     }
587   }
588   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
589   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
590 }

```

(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

```

591 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
592   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
593     \str_set:Nx \l_tmpa_str { #1 }
594     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
595     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
596     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
597     \_stex_mathhub_find_manifest:N \l_tmpa_seq
598     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
599       \msg_error:nnxx{stex}{error/norepository}{#1}{
600         \stex_path_to_string:N \c_stex_mathhub_str
601       }
602       \input{Fatal-Error!}
603     } {
604       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
605     }
606   }
607 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

\l_stex_mathhub_manifest_file_seq

608 \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:

```

609 \cs_new_protected:Nn \__stex_mathhub_find_manifest:N {
610   \seq_set_eq:NN\l_tmpa_seq #1
611   \bool_set_true:N\l_tmpa_bool
612   \bool_while_do:Nn \l_tmpa_bool {
613     \seq_if_empty:NTF \l_tmpa_seq {
614       \bool_set_false:N\l_tmpa_bool
615     }{
616       \file_if_exist:nTF{
617         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
618       }{
619         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
620         \bool_set_false:N\l_tmpa_bool
621       }{
622         \file_if_exist:nTF{
623           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
624         }{
625           \seq_put_right:Nn\l_tmpa_seq{META-INF}
626           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
627           \bool_set_false:N\l_tmpa_bool
628         }{
629           \file_if_exist:nTF{
630             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
631           }{
632             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
633             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
634             \bool_set_false:N\l_tmpa_bool
635           }{
636             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
637           }
638         }
639       }
640     }
641   }
642   \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
643 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c_stex_mathhub_manifest_ior

File variable used for MANIFEST-files

644 \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:

```

645 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
646   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
647   \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}

```

```

648 \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
649   \str_set:Nn \l_tmpa_str {##1}
650   \exp_args:NNoo \seq_set_split:Nnn
651     \l_tmpb_seq \c_colon_str \l_tmpa_str
652   \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
653     \exp_args:NNe \str_set:Nn \l_tmpb_tl {
654       \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
655     }
656     \exp_args:No \str_case:nnTF \l_tmpa_tl {
657       {id} {
658         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
659           { id } \l_tmpb_tl
660       }
661       {narration-base} {
662         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
663           { narr } \l_tmpb_tl
664       }
665       {url-base} {
666         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
667           { docurl } \l_tmpb_tl
668       }
669       {source-base} {
670         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
671           { ns } \l_tmpb_tl
672       }
673       {ns} {
674         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
675           { ns } \l_tmpb_tl
676       }
677       {dependencies} {
678         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
679           { deps } \l_tmpb_tl
680       }
681     }{}{}
682   }{}
683 }
684 \ior_close:N \c__stex_mathhub_manifest_ior
685 \stex_persist:x {
686   \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
687     \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
688   }
689 }
690 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

691 \cs_new_protected:Nn \stex_set_current_repository:n {
692   \stex_require_repository:n { #1 }
693   \prop_set_eq:Nc \l_stex_current_repository_prop {
694     c_stex_mathhub_#1_manifest_prop
695   }
696 }

```

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

`\stex_require_repository:n`

```

697 \cs_new_protected:Nn \stex_require_repository:n {
698   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
699     \stex_debug:nn{mathhub}{Opening~archive:~#1}
700     \__stex_mathhub_do_manifest:n { #1 }
701   }
702 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

703 %\prop_new:N \l_stex_current_repository_prop
704 \bool_if:NF \c_stex_persist_mode_bool {
705   \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
706   \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
707     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
708   } {
709     \__stex_mathhub_parse_manifest:n { main }
710     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
711     \l_tmpa_str
712     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
713     \c_stex_mathhub_main_manifest_prop
714     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
715     \stex_debug:nn{mathhub}{Current~repository:~
716     \prop_item:Nn \l_stex_current_repository_prop {id}
717   }
718 }
719 }

```

(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.

```

720 \cs_new_protected:Nn \stex_in_repository:nn {
721   \str_set:Nx \l_tmpa_str { #1 }
722   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
723   \str_if_empty:NTF \l_tmpa_str {
724     \prop_if_exist:NTF \l_stex_current_repository_prop {
725       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
726       \exp_args:Ne \l_tmpa_cs{
727         \prop_item:Nn \l_stex_current_repository_prop { id }
728       }
729     }{
730       \l_tmpa_cs{}
731     }
732   }{
733     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
734     \stex_require_repository:n \l_tmpa_str
735     \str_set:Nx \l_tmpa_str { #1 }
736     \exp_args:Nne \use:nn {
737       \stex_set_current_repository:n \l_tmpa_str
738       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
739     }{
740       \stex_debug:nn{mathhub}{switching~back~to:~

```

```

741     \prop_if_exist:NTF \l_stex_current_repository_prop {
742       \prop_item:Nn \l_stex_current_repository_prop { id }::~
743       \meaning\l_stex_current_repository_prop
744     }{
745       no~repository
746     }
747   }
748   \prop_if_exist:NTF \l_stex_current_repository_prop {
749     \stex_set_current_repository:n {
750       \prop_item:Nn \l_stex_current_repository_prop { id }
751     }
752   }{
753     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
754   }
755 }
756 }
757 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

758 \def \mhpath #1 #2 {
759   \exp_args:Ne \tl_if_empty:nTF{#1}{
760     \c_stex_mathhub_str /
761     \prop_item:Nn \l_stex_current_repository_prop { id }
762     / source / #2
763   }{
764     \c_stex_mathhub_str / #1 / source / #2
765   }
766 }

```

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

`\inputref`

`\mhinput`

```

767 \newif \ifinputref \inputreffalse
768
769 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
770   \stex_in_repository:nn {#1} {
771     \ifinputref
772       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
773     \else
774       \inputreftrue
775       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
776       \inputreffalse
777     \fi
778   }
779 }
780 \NewDocumentCommand \mhinput { 0{} m }{
781   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
782 }
783

```

```

784 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
785   \stex_in_repository:nn {#1} {
786     \stex_html_backend:TF {
787       \str_clear:N \l_tmpa_str
788       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
789         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
790       }
791
792       \tl_if_empty:nTF{ ##1 }{
793         \IfFileExists{#2}{
794           \stex_annotate_invisible:nnn{inputref}{
795             \l_tmpa_str / #2
796           }{}
797         }{
798           \input{#2}
799         }
800       }{
801         \IfFileExists{ \c_stex_mathhub_str / ##1 / source / #2 }{
802           \stex_annotate_invisible:nnn{inputref}{
803             \l_tmpa_str / #2
804           }{}
805         }{
806           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
807         }
808       }
809
810     }{
811       \begingroup
812       \inputreftrue
813       \tl_if_empty:nTF{ ##1 }{
814         \input{#2}
815       }{
816         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
817       }
818       \endgroup
819     }
820   }
821 }
822 \NewDocumentCommand \inputref { 0{} m}{
823   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
824 }

```

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

`\addmhbibresource`

```

825 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
826   \stex_in_repository:nn {#1} {
827     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
828   }
829 }
830 \newcommand\addmhbibresource[2][]{
831   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
832 }

```

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

`\libinput`

```
833 \cs_new_protected:Npn \libinput #1 {
834   \prop_if_exist:NF \l_stex_current_repository_prop {
835     \msg_error:nnn{stex}{error/notinarchive}\libinput
836   }
837   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
838     \msg_error:nnn{stex}{error/notinarchive}\libinput
839   }
840   \seq_clear:N \l__stex_mathhub_libinput_files_seq
841   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
842   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
843
844   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
845     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
846     \IfFileExists{ \l_tmpa_str }{
847       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
848     }{}
849     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
850     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
851   }
852
853   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
854   \IfFileExists{ \l_tmpa_str }{
855     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
856   }{}
857
858   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
859     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
860   }{
861     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
862       \input{ ##1 }
863     }
864   }
865 }
```

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

`\libusepackage`

```
866 \NewDocumentCommand \libusepackage {0{ } m} {
867   \prop_if_exist:NF \l_stex_current_repository_prop {
868     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
869   }
870   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
871     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
872   }
873   \seq_clear:N \l__stex_mathhub_libinput_files_seq
874   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
875   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
876
877   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
878     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
879     \IfFileExists{ \l_tmpa_str.sty }{
880       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
881     }{}
882   }
```

```

882 \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
883 \seq_put_right:No \l_tmpa_seq \l_tmpa_str
884 }
885
886 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
887 \IfFileExists{ \l_tmpa_str.sty }{
888 \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
889 }{}
890
891 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
892 \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
893 }{
894 \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
895 \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
896 \usepackage[#1]{ #1 }
897 }
898 }{
899 \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
900 }
901 }
902 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

903
904 \AddToHook{begindocument}{
905 \ltx@ifpackageloaded{graphicx}{
906 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
907 \providecommand\mhgraphics[2] [] {%
908 \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
909 \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
910 \providecommand\cmhgraphics[2] [] {\begin{center}\mhgraphics[#1]{#2}\end{center}}
911 }{}

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

912 \ltx@ifpackageloaded{listings}{
913 \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
914 \newcommand\lstinputmhlisting[2] [] {%
915 \def\lst@mhrepos{}\setkeys{lst}{#1}%
916 \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
917 \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
918 }{}
919 }
920
921 </package>

```

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

Chapter 26

STEX -References Implementation

```
922 <*package>
923
924 %%%%%%%%%% references.dtx %%%%%%%%%%
925
926 <@@=stex_refs>
    Warnings and error messages
927
```

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

```
928 %\iow_new:N \c__stex_refs_refs_iow
929 \AtBeginDocument{
930 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
931 }
932 \AtEndDocument{
933 % \iow_close:N \c__stex_refs_refs_iow
934 }
```

`\STEXreftitle`

```
935 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
936
937 \NewDocumentCommand \STEXreftitle { m } {
938   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
939 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
940 \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:`

```
941 \cs_new_protected:Nn \stex_get_document_uri: {
942   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
943   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
944   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
945   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
946   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
947
948   \str_clear:N \l_tmpa_str
949   \prop_if_exist:NT \l_stex_current_repository_prop {
950     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
951       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
952     }
953   }
954
955   \str_if_empty:NTF \l_tmpa_str {
956     \str_set:Nx \l_stex_current_docns_str {
957       file:/\stex_path_to_string:N \l_tmpa_seq
958     }
959   }{
960     \bool_set_true:N \l_tmpa_bool
961     \bool_while_do:Nn \l_tmpa_bool {
962       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
963       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
964         {source} { \bool_set_false:N \l_tmpa_bool }
965       }{}{
966         \seq_if_empty:NT \l_tmpa_seq {
967           \bool_set_false:N \l_tmpa_bool
968         }
969       }
970     }
971
972     \seq_if_empty:NTF \l_tmpa_seq {
973       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
974     }{
975       \str_set:Nx \l_stex_current_docns_str {
976         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
977       }
978     }
979   }
980 }
```

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

`\l_stex_current_docurl_str`

```
981 \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:`

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

```

986 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
987 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
988
989 \str_clear:N \l_tmpa_str
990 \prop_if_exist:NT \l_stex_current_repository_prop {
991   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
992     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
993       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
994     }
995   }
996 }
997
998 \str_if_empty:NTF \l_tmpa_str {
999   \str_set:Nx \l_stex_current_docurl_str {
1000     file:/\stex_path_to_string:N \l_tmpa_seq
1001   }
1002 }{
1003   \bool_set_true:N \l_tmpa_bool
1004   \bool_while_do:Nn \l_tmpa_bool {
1005     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1006     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1007       {source} { \bool_set_false:N \l_tmpa_bool }
1008     }{}{
1009       \seq_if_empty:NT \l_tmpa_seq {
1010         \bool_set_false:N \l_tmpa_bool
1011       }
1012     }
1013   }
1014 }
1015
1016 \seq_if_empty:NTF \l_tmpa_seq {
1017   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
1018 }{
1019   \str_set:Nx \l_stex_current_docurl_str {
1020     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
1021   }
1022 }
1023 }

```

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

26.2 Setting Reference Targets

```

1024 \str_const:Nn \c__stex_refs_url_str{URL}
1025 \str_const:Nn \c__stex_refs_ref_str{REF}
1026 \str_new:N \l__stex_refs_curr_label_str
1027 % @currentlabel -> number
1028 % @currentlabelname -> title
1029 % @currentHref -> name.number <- id of some kind
1030 % \theH# -> \arabic{section}
1031 % \the# -> number
1032 % \hyper@makecurrent{#}
1033 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

1034 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
1035   \stex_get_document_uri:
1036   \str_clear:N \l__stex_refs_curr_label_str
1037   \str_set:Nx \l_tmpa_str { #1 }
1038   \str_if_empty:NT \l_tmpa_str {
1039     \int_incr:N \l__stex_refs_unnamed_counter_int
1040     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
1041   }
1042   \str_set:Nx \l__stex_refs_curr_label_str {
1043     \l_stex_current_docns_str?\l_tmpa_str
1044   }
1045   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
1046     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
1047   }
1048   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
1049     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
1050   }
1051   \stex_if_smsmode:TF {
1052     \stex_get_document_url:
1053     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
1054     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
1055   }{
1056     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
1057     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
1058     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
1059     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
1060   }
1061 }

```

(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.

```

1062 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
1063   \str_set:Nn \l_tmpa_str {#1?#2}
1064   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
1065   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
1066     \seq_new:c {g__stex_refs_labels_#2_seq}
1067   }
1068   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
1069     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
1070   }
1071 }

```

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

```

1072 \AtEndDocument{
1073   \def\stexauxadddocref#1 #2 {}{}
1074 }

```

`\stex_ref_new_sym_target:n`

```

1075 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
1076   \stex_if_smsmode:TF {
1077     \str_if_exist:cF{sref_sym_#1_type}{
1078       \stex_get_document_url:
1079       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

1080     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
1081   }
1082   ){
1083     \str_if_empty:NF \l__stex_refs_curr_label_str {
1084       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
1085       \immediate\write\@auxout{
1086         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
1087           \l__stex_refs_curr_label_str
1088         }
1089       }
1090     }
1091   }
1092 }

```

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

26.3 Using References

```

1093 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

1094
1095 \keys_define:nn { stex / sref } {
1096   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
1097   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
1098   pre           .tl_set:N = \l__stex_refs_pre_tl ,
1099   post          .tl_set:N = \l__stex_refs_post_tl ,
1100 }
1101 \cs_new_protected:Nn \__stex_refs_args:n {
1102   \tl_clear:N \l__stex_refs_linktext_tl
1103   \tl_clear:N \l__stex_refs_fallback_tl
1104   \tl_clear:N \l__stex_refs_pre_tl
1105   \tl_clear:N \l__stex_refs_post_tl
1106   \str_clear:N \l__stex_refs_repo_str
1107   \keys_set:nn { stex / sref } { #1 }
1108 }

```

The actual macro:

```

1109 \NewDocumentCommand \sref { 0{} m}{
1110   \__stex_refs_args:n { #1 }
1111   \str_if_empty:NTF \l__stex_refs_indocument_str {
1112     \str_set:Nx \l_tmpa_str { #2 }
1113     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1114     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1115       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1116         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1117           \str_clear:N \l_tmpa_str
1118         }
1119       }{
1120         \str_clear:N \l_tmpa_str
1121       }
1122     }{
1123       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1124       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

1125 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1126 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1127   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1128   \str_clear:N \l_tmpa_str
1129   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1130     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1131       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1132     }{
1133       \seq_map_break:n {
1134         \str_set:Nn \l_tmpa_str { ##1 }
1135       }
1136     }
1137   }
1138 }{
1139   \str_clear:N \l_tmpa_str
1140 }
1141 }
1142 \str_if_empty:NTF \l_tmpa_str {
1143   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1144 }{
1145   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1146     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1147       \cs_if_exist:cTF{autoref}{
1148         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1149       }{
1150         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1151       }
1152     }{
1153       \ltx@ifpackageloaded{hyperref}{
1154         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1155       }{
1156         \l__stex_refs_linktext_tl
1157       }
1158     }
1159   }{
1160     \ltx@ifpackageloaded{hyperref}{
1161       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1162     }{
1163       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1164     }
1165   }
1166 }
1167 }{
1168   % TODO
1169 }
1170 }

```

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

\srefsym

```

1171 \NewDocumentCommand \srefsym { 0{} m}{
1172   \stex_get_symbol:n { #2 }
1173   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1174 }

```

```

1175
1176 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1177   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1178     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1179   }{
1180     \__stex_refs_args:n { #1 }
1181     \str_if_empty:NTF \l__stex_refs_indocument_str {
1182       \tl_if_exist:cTF{sref_sym_#2 _type}{
1183         % doc uri in \l_tmpb_str
1184         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1185         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1186           % reference
1187           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1188             \cs_if_exist:cTF{autoref}{
1189               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1190             }{
1191               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1192             }
1193           }{
1194             \ltx@ifpackageloaded{hyperref}{
1195               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1196             }{
1197               \l__stex_refs_linktext_tl
1198             }
1199           }
1200         }{
1201           % URL
1202           \ltx@ifpackageloaded{hyperref}{
1203             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1204           }{
1205             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1206           }
1207         }
1208       }{
1209         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1210       }
1211     }{
1212       % TODO
1213     }
1214   }
1215 }

```

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

\srefsymuri

```

1216 \cs_new_protected:Npn \srefsymuri #1 #2 {
1217   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1218 }

```

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

```

1219 </package>

```

Chapter 27

STEX -Modules Implementation

```
1220 <*package>
1221
1222 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1223
1224 <@@=stex_modules>
1225
1226 Warnings and error messages
1227 \msg_new:nnn{stex}{error/unknownmodule}{
1228   No~module~#1~found
1229 }
1230 \msg_new:nnn{stex}{error/syntax}{
1231   Syntax~error:~#1
1232 }
1233 \msg_new:nnn{stex}{error/siglanguage}{
1234   Module~#1~declares~signature~#2,~but~does~not~
1235   declare~its~language
1236 }
1237 \msg_new:nnn{stex}{warning/deprecated}{
1238   #1~is~deprecated;~please~use~#2~instead!
1239 }
1240 \msg_new:nnn{stex}{error/conflictingmodules}{
1241   Conflicting~imports~for~module~#1
1242 }
```

```
\l_stex_current_module_str The current module:
1242 \str_new:N \l_stex_current_module_str
(End definition for \l_stex_current_module_str. This variable is documented on page 71.)
```

```
\l_stex_all_modules_seq Stores all available modules
1243 \seq_new:N \l_stex_all_modules_seq
(End definition for \l_stex_all_modules_seq. This variable is documented on page 71.)
```



```

\stex_if_in_module_p:
\stex_if_in_module:TF
1244 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1245   \str_if_empty:NTF \l_stex_current_module_str
1246   \prg_return_false: \prg_return_true:
1247 }

(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
1248 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1249   \prop_if_exist:cTF { c_stex_module_#1_prop }
1250   \prg_return_true: \prg_return_false:
1251 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
1252 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1253   \stex_add_to_current_module:n { #1 }
1254   \stex_do_up_to_module:n { #1 }
1255 }}
1256 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1257
1258 \cs_new_protected:Nn \stex_add_to_current_module:n {
1259   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str_code} { #1 }
1260 }
1261 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1262 \cs_new_protected:Npn \STEXexport {
1263   \ExplSyntaxOn
1264   \__stex_modules_export:n
1265 }
1266 \cs_new_protected:Nn \__stex_modules_export:n {
1267   \ignorespacesandpars#1\ExplSyntaxOff
1268   \stex_add_to_current_module:n { \ignorespacesandpars#1}
1269   \stex_smsmode_do:
1270 }
1271 \let \stex_module_export_helper:n \use:n
1272 \stex_deactivate_macro:Nn \STEXexport {module~environments}

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

```

```

\stex_add_constant_to_current_module:n
1273 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1274   \str_set:Nx \l_tmpa_str { #1 }
1275   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str_constants} { \l_tmpa_str }
1276 }

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

```

```

\stex_add_import_to_current_module:n
1277 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1278   \str_set:Nx \l_tmpa_str { #1 }
1279   \exp_args:Nno

```

```

1280 \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1281 \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1282 }
1283 }

```

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

`\stex_collect_imports:n`

```

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

```

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

`\stex_do_up_to_module:n`

```

1298 \int_new:N \l__stex_modules_group_depth_int
1299 \cs_new_protected:Nn \stex_do_up_to_module:n {
1300 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1301 #1
1302 }{
1303 #1
1304 \expandafter \tl_gset:Nn
1305 \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1306 \expandafter\expandafter\expandafter\endcsname
1307 \expandafter\expandafter\expandafter { \csname
1308 l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1309 \aftergroup\__stex_modules_aftergroup_do:
1310 }
1311 }
1312 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1313 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1314 \stex_debug:nn{aftergroup}{\cs_meaning:c{
1315 l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1316 }}}
1317 \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1318 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1319 \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1320 }{
1321 \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1322 \aftergroup\__stex_modules_aftergroup_do:
1323 }
1324 }
1325 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1326 \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined

```

1327 }

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

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

1328

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

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```
1329 \str_new:N \l_stex_module_ns_str
1330 \str_new:N \l_stex_module_subpath_str
1331 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1332   \seq_set_eq:NN \l_tmpa_seq #2
1333   % split off file extension
1334   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1335   \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1336   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1337   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1338
1339   \bool_set_true:N \l_tmpa_bool
1340   \bool_while_do:Nn \l_tmpa_bool {
1341     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1342     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1343       {source} { \bool_set_false:N \l_tmpa_bool }
1344     }{}{
1345       \seq_if_empty:NT \l_tmpa_seq {
1346         \bool_set_false:N \l_tmpa_bool
1347       }
1348     }
1349   }
1350
1351   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1352   % \l_tmpa_seq <- sub-path relative to archive
1353   \str_if_empty:NTF \l_stex_module_subpath_str {
1354     \str_set:Nx \l_stex_module_ns_str {#1}
1355   }{
1356     \str_set:Nx \l_stex_module_ns_str {
1357       #1/\l_stex_module_subpath_str
1358     }
1359   }
1360 }
1361
1362 \cs_new_protected:Nn \stex_modules_current_namespace: {
1363   \str_clear:N \l_stex_module_subpath_str
1364   \prop_if_exist:NTF \l_stex_current_repository_prop {
1365     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1366     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1367   }{
1368     % split off file extension
1369     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1370     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

1371 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1372 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1373 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1374 \str_set:Nx \l_stex_module_ns_str {
1375   file:/\stex_path_to_string:N \l_tmpa_seq
1376 }
1377 }
1378 }

```

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

27.1 The smodule environment

smodule arguments:

```

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

```

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

```

1410 \cs_new_protected:Nn \stex_module_setup:nn {
1411   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1412   \str_set:Nx \l_stex_module_name_str { #2 }
1413   \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.

Are we in a nested module?

```

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

```

Next, we determine the language of the module:

```

1442 \str_if_empty:NT \l_stex_module_lang_str {
1443   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1444   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1445   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1446   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1447     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1448       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1449     }
1450   }
1451   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1452   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1453     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1454     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1455       inferred~from~file~name}
1456   }
1457 }
1458
1459 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1460   \exp_args:NNo \stex_set_language:Nn \l_tmpa_str \l_stex_module_lang_str
1461 }}

```

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

```

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

```

We load the metatheory:

```

1479 \str_if_empty:NT \l_stex_module_meta_str {
1480   \str_set:Nx \l_stex_module_meta_str {
1481     \c_stex_metatheory_ns_str ? Metatheory
1482   }
1483 }
1484 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1485   \bool_set_true:N \l_stex_in_meta_bool
1486   \exp_args:Nx \stex_add_to_current_module:n {
1487     \bool_set_true:N \l_stex_in_meta_bool
1488     \stex_activate_module:n {\l_stex_module_meta_str}
1489     \bool_set_false:N \l_stex_in_meta_bool
1490   }
1491   \stex_activate_module:n {\l_stex_module_meta_str}
1492   \bool_set_false:N \l_stex_in_meta_bool
1493 }
1494 }{
1495   \str_if_empty:NT \l_stex_module_lang_str {
1496     \msg_error:nnxx{stex}{error/siglanguage}{
1497       \l_stex_module_ns_str?\l_stex_module_name_str
1498     }{\l_stex_module_sig_str}
1499   }
1500   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1501   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1502     \stex_debug:nn{modules}{(already exists)}
1503   }{
1504     \stex_debug:nn{modules}{(needs loading)}
1505     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1506     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1507     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1508     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1509     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1510     \str_set:Nx \l_tmpa_str {
1511       \stex_path_to_string:N \l_tmpa_seq /

```

```

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

```

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

smodule The module environment.

```

\__stex_modules_begin_module: implements \begin{smodule}
1543 \cs_new_protected:Nn \__stex_modules_begin_module: {
1544     \stex_reactivate_macro:N \STEXexport
1545     \stex_reactivate_macro:N \importmodule
1546     \stex_reactivate_macro:N \symdecl
1547     \stex_reactivate_macro:N \notation
1548     \stex_reactivate_macro:N \symdef
1549
1550     \stex_debug:nn{modules}{
1551         New~module:\\
1552         Namespace:~\l_stex_module_ns_str\\
1553         Name:~\l_stex_module_name_str\\
1554         Language:~\l_stex_module_lang_str\\
1555         Signature:~\l_stex_module_sig_str\\
1556         Metatheory:~\l_stex_module_meta_str\\
1557         File:~\stex_path_to_string:N \g_stex_currentfile_seq
1558     }
1559

```

```

1560 \stex_if_do_html:T{
1561   \begin{stex_annotate_env} {theory} {
1562     \l_stex_module_ns_str ? \l_stex_module_name_str
1563   }
1564
1565   \stex_annotate_invisible:nnn{header}{} {
1566     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1567     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1568     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1569       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1570     }
1571     \str_if_empty:NF \smoduletype {
1572       \stex_annotate:nnn{type}{\smoduletype}{}
1573     }
1574   }
1575 }
1576 % TODO: Inherit metatheory for nested modules?
1577 }
1578 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `_stex_modules_begin_module:.`)

`_stex_modules_end_module:` implements `\end{module}`

```

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

```

(End definition for `_stex_modules_end_module:.`)

The core environment

```

1599 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1600 \NewDocumentEnvironment { smodule } { 0 } { m } {
1601   \stex_module_setup:nn{#1}{#2}
1602   %\par
1603   \stex_if_smsmode:F{
1604     \tl_if_empty:NF \smoduletitle {
1605       \exp_args:No \stex_document_title:n \smoduletitle
1606     }

```



```

1607 \tl_clear:N \l_tmpa_tl
1608 \clist_map_inline:Nn \smodulotype {
1609   \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1610     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1611   }
1612 }
1613 \tl_if_empty:NTF \l_tmpa_tl {
1614   \__stex_modules_smodule_start:
1615 }{
1616   \l_tmpa_tl
1617 }
1618 }
1619 \__stex_modules_begin_module:
1620 \str_if_empty:NF \smoduleid {
1621   \stex_ref_new_doc_target:n \smoduleid
1622 }
1623 \stex_smsmode_do:
1624 } {
1625   \__stex_modules_end_module:
1626   \stex_if_smsmode:F {
1627     \end{stex_annotate_env}
1628     \clist_set:Nn \l_tmpa_clist \smodulotype
1629     \tl_clear:N \l_tmpa_tl
1630     \clist_map_inline:Nn \l_tmpa_clist {
1631       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1632         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1633       }
1634     }
1635     \tl_if_empty:NTF \l_tmpa_tl {
1636       \__stex_modules_smodule_end:
1637     }{
1638       \l_tmpa_tl
1639     }
1640   }
1641 }

```

\stexpatchmodule

```

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

```

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

27.2 Invoking modules

```

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

```

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

```

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

```

```

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

```

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

```

1708 \endpackage

```

Chapter 28

STEX -Module Inheritance Implementation

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

28.1 SMS Mode

```
1713 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

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

```

1737 \copynotation
1738 \assign
1739 \renamedekl
1740 \donotcopy
1741 \instantiate
1742 \textsymdecl
1743 }
1744
1745 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1746   \tl_to_str:n {
1747     smodule,
1748     copymodule,
1749     interpretmodule,
1750     realization,
1751     sdefinition,
1752     sexample,
1753     sassertion,
1754     sparagraph,
1755     mathstructure
1756   }
1757 }

```

(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
1758 \bool_new:N \g__stex_smsmode_bool
1759 \bool_set_false:N \g__stex_smsmode_bool
1760 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1761   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1762 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1763 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1764   \vbox_set:Nn \l_tmpa_box {
1765     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1766     \bool_gset_true:N \g__stex_smsmode_bool
1767     #2
1768     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1769   }
1770   \box_clear:N \l_tmpa_box
1771 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1772 \quark_new:N \q__stex_smsmode_break
1773
1774 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1775   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1776   \stex_smsmode_do:
1777 }
1778

```

```

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

```

```

1833 \let\__stex_modules_begin_module:\relax
1834 \let\__stex_modules_end_module:\relax
1835 \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1836 \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{smodule}}
1837 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1838 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1839 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1840 \everyeof{\q__stex_smsmode_break\noexpand}
1841 \expandafter\expandafter\expandafter
1842 \stex_smsmode_do:
1843 \csname @ @ input\endcsname "#1"\relax
1844
1845 \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1846   \stex_filestack_push:n{##1}
1847   \expandafter\expandafter\expandafter
1848   \stex_smsmode_do:
1849   \csname @ @ input\endcsname "##1"\relax
1850   \stex_filestack_pop:
1851 }
1852 }
1853 % ----- new -----
1854 \__stex_smsmode_in_smsmode:nn{#1} {
1855   #2
1856   % ----- new -----
1857   \begingroup
1858   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1859   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1860     \__stex_smsmode_check_import_pair:nnT ##1 { \begingroup
1861       \stex_import_module_uri:nn ##1
1862       \stex_import_require_module:nnnn
1863       \l_stex_import_ns_str
1864       \l_stex_import_archive_str
1865       \l_stex_import_path_str
1866       \l_stex_import_name_str \endgroup
1867     }
1868   }
1869   \endgroup
1870   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1871   % ----- new -----
1872   \everyeof{\q__stex_smsmode_break\noexpand}
1873   \expandafter\expandafter\expandafter
1874   \stex_smsmode_do:
1875   \csname @ @ input\endcsname "#1"\relax
1876 }
1877 \stex_filestack_pop:
1878 }

```

(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:

```

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

```

```

1882 }
1883 }
1884 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1885   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1886     \expandafter\if\expandafter\relax\noexpand#1
1887     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1888   \else\expandafter\__stex_smsmode_do:w\fi
1889 }{
1890   \__stex_smsmode_do:w % #1
1891 }
1892 }
1893 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1894   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1895     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1896       #1\__stex_smsmode_do:w
1897     }{
1898       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1899         #1
1900       }{
1901         \cs_if_eq:NNTF \begin #1 {
1902           \__stex_smsmode_check_begin:n
1903         }{
1904           \cs_if_eq:NNTF \end #1 {
1905             \__stex_smsmode_check_end:n
1906           }{
1907             \__stex_smsmode_do:w
1908           }
1909         }
1910       }
1911     }
1912   }
1913 }
1914
1915 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1916   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1917     \begin{#1}
1918   }{
1919     \__stex_smsmode_do:w
1920   }
1921 }
1922 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1923   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1924     \end{#1}\__stex_smsmode_do:w
1925   }{
1926     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1927   }
1928 }

```

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

28.2 Inheritance

```

1929 <@@=stex_importmodule>

```


`\stex_import_module_uri:nn`

```
1930 \cs_new_protected:Nn \stex_import_module_uri:nn {
1931   \str_set:Nx \l_stex_import_archive_str { #1 }
1932   \str_set:Nn \l_stex_import_path_str { #2 }
1933
1934   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1935   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1936   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1937
1938   \stex_modules_current_namespace:
1939   \bool_lazy_all:nTF {
1940     {\str_if_empty_p:N \l_stex_import_archive_str}
1941     {\str_if_empty_p:N \l_stex_import_path_str}
1942     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1943   }{
1944     \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1945     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1946   }{
1947     \str_if_empty:NT \l_stex_import_archive_str {
1948       \prop_if_exist:NT \l_stex_current_repository_prop {
1949         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1950       }
1951     }
1952     \str_if_empty:NTF \l_stex_import_archive_str {
1953       \str_if_empty:NF \l_stex_import_path_str {
1954         \stex_path_from_string:Nn \l_tmpb_seq {
1955           \l_stex_module_ns_str / .. / \l_stex_import_path_str
1956         }
1957         \str_set:Nx \l_stex_import_ns_str {\stex_path_to_string:N \l_tmpb_seq}
1958         \str_replace_once:Nnn \l_stex_import_ns_str {file://} {file:///}
1959       }
1960     }{
1961       \stex_require_repository:n \l_stex_import_archive_str
1962       \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str_manifest_prop } { ns }
1963       \l_stex_import_ns_str
1964       \str_if_empty:NF \l_stex_import_path_str {
1965         \str_set:Nx \l_stex_import_ns_str {
1966           \l_stex_import_ns_str / \l_stex_import_path_str
1967         }
1968       }
1969     }
1970   }
1971 }
```

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

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

Store the return values of `\stex_import_module_uri:nn`.

```
1972 \str_new:N \l_stex_import_name_str
1973 \str_new:N \l_stex_import_archive_str
1974 \str_new:N \l_stex_import_path_str
1975 \str_new:N \l_stex_import_ns_str
```

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

```

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

```

```

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

```

```

2083     \str_clear:N \l_stex_current_module_str
2084     \str_set:Nx \l_tmpb_str { #2 }
2085     \str_if_empty:NF \l_tmpb_str {
2086       \stex_set_current_repository:n { #2 }
2087     }
2088     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
2089   }
2090
2091   \stex_if_module_exists:nF { #1 ? #4 } {
2092     \msg_error:nnx{stex}{error/unknownmodule}{
2093       #1?#4~(in~file~\g__stex_importmodule_file_str)
2094     }
2095   }
2096 }
2097
2098 }
2099 \stex_activate_module:n { #1 ? #4 }
2100 }

```

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

`\importmodule`

```

2101 \NewDocumentCommand \importmodule { 0{} m } {
2102   \stex_import_module_uri:nn { #1 } { #2 }
2103   \stex_debug:nn{modules}{Importing~module:~
2104     \l_stex_import_ns_str ? \l_stex_import_name_str
2105   }
2106   \stex_import_require_module:nnnn
2107   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2108   { \l_stex_import_path_str } { \l_stex_import_name_str }
2109   \stex_if_smsmode:F {
2110     \stex_annotate_invisible:nnn
2111     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
2112   }
2113   \exp_args:Nx \stex_add_to_current_module:n {
2114     \stex_import_require_module:nnnn
2115     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
2116     { \l_stex_import_path_str } { \l_stex_import_name_str }
2117   }
2118   \exp_args:Nx \stex_add_import_to_current_module:n {
2119     \l_stex_import_ns_str ? \l_stex_import_name_str
2120   }
2121   \stex_smsmode_do:
2122   \ignorespacesandpars
2123 }
2124 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

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

```

```

2130 { \l_stex_import_path_str } { \l_stex_import_name_str }
2131 \stex_annotate_invisible:nnn
2132 {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
2133 }
2134 \stex_smsmode_do:
2135 \ignorespacesandpars
2136 }

```

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

```

2137 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
2138   \tl_if_empty:nF{#2}{
2139     \clist_set:Nn \l_tmpa_clist {#2}
2140     \clist_map_inline:Nn \l_tmpa_clist {
2141       \tl_if_head_eq_charcode:nNTF {##1} [{
2142         #1 ##1
2143       }{
2144         #1{##1}
2145       }
2146     }
2147   }
2148 }
2149 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2150
2151
2152 \endpackage

```

Chapter 29

STEX -Symbols Implementation

```
2153 <*package>
2154
2155 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2156
    Warnings and error messages
2157 \msg_new:nnn{stex}{error/wrongargs}{
2158   args~value~in~symbol~declaration~for~#1~
2159   needs~to~be~i,~a,~b~or~B,~but~#2~given
2160 }
2161 \msg_new:nnn{stex}{error/unknownsymbol}{
2162   No~symbol~#1~found!
2163 }
2164 \msg_new:nnn{stex}{error/seqlength}{
2165   Expected~#1~arguments;~got~#2!
2166 }
2167 \msg_new:nnn{stex}{error/unknownnotation}{
2168   Unknown~notation~#1~for~#2!
2169 }
```

29.1 Symbol Declarations

```
2170 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2171 \cs_new_protected:Nn \stex_all_symbols:n {
2172   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2173   \seq_map_inline:Nn \l_stex_all_modules_seq {
2174     \seq_map_inline:cn{c_stex_module_##1_constants}{
2175       \__stex_symdecl_all_symbols_cs{##1?####1}
2176     }
2177   }
2178 }
```

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

`\STEXsymbol`

```
2179 \NewDocumentCommand \STEXsymbol { m } {  
2180   \stex_get_symbol:n { #1 }  
2181   \exp_args:No  
2182   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }  
2183 }
```

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

`symdecl` arguments:

```
2184 \keys_define:nn { stex / symdecl } {  
2185   name      .str_set:x:N = \l_stex_symdecl_name_str ,  
2186   local     .bool_set:N = \l_stex_symdecl_local_bool ,  
2187   args      .str_set:x:N = \l_stex_symdecl_args_str ,  
2188   type      .tl_set:N = \l_stex_symdecl_type_tl ,  
2189   deprecate .str_set:x:N = \l_stex_symdecl_deprecate_str ,  
2190   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)  
2191   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)  
2192   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)  
2193   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,  
2194   reorder   .str_set:x:N = \l_stex_symdecl_reorder_str ,  
2195   assoc     .choices:nn =  
2196     {bin,binl,binr,pre,conj,pwconj}  
2197     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}  
2198 }  
2199  
2200 \bool_new:N \l_stex_symdecl_make_macro_bool  
2201  
2202 \cs_new_protected:Nn \__stex_symdecl_args:n {  
2203   \str_clear:N \l_stex_symdecl_name_str  
2204   \str_clear:N \l_stex_symdecl_args_str  
2205   \str_clear:N \l_stex_symdecl_deprecate_str  
2206   \str_clear:N \l_stex_symdecl_reorder_str  
2207   \str_clear:N \l_stex_symdecl_assoctype_str  
2208   \bool_set_false:N \l_stex_symdecl_local_bool  
2209   \tl_clear:N \l_stex_symdecl_type_tl  
2210   \tl_clear:N \l_stex_symdecl_definiens_tl  
2211  
2212   \keys_set:nn { stex / symdecl } { #1 }  
2213 }
```

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

```
2214  
2215 \NewDocumentCommand \symdecl { s m O{} } {  
2216   \__stex_symdecl_args:n { #3 }  
2217   \IfBooleanTF #1 {  
2218     \bool_set_false:N \l_stex_symdecl_make_macro_bool  
2219   } {  
2220     \bool_set_true:N \l_stex_symdecl_make_macro_bool  
2221   }  
2222   \stex_symdecl_do:n { #2 }  
2223   \stex_smsmode_do:  
2224 }
```

```

2225
2226 \cs_new_protected:Nn \stex_symdecl_do:nn {
2227   \__stex_symdecl_args:n{#1}
2228   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2229   \stex_symdecl_do:n{#2}
2230 }
2231
2232 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2233 \cs_new_protected:Nn \stex_symdecl_do:n {
2234   \stex_if_in_module:F {
2235     % TODO throw error? some default namespace?
2236   }
2237
2238   \str_if_empty:NT \l_stex_symdecl_name_str {
2239     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2240   }
2241
2242   \prop_if_exist:cT { l_stex_symdecl_
2243     \l_stex_current_module_str ?
2244     \l_stex_symdecl_name_str
2245     _prop
2246   }{
2247     % TODO throw error (beware of circular dependencies)
2248   }
2249
2250   \prop_clear:N \l_tmpa_prop
2251   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2252   \seq_clear:N \l_tmpa_seq
2253   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2254   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2255
2256   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2257     \str_if_empty:NF \l_stex_module_deprecate_str {
2258       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2259     }
2260   }
2261   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2262
2263   \exp_args:No \stex_add_constant_to_current_module:n {
2264     \l_stex_symdecl_name_str
2265   }
2266
2267   % arity/args
2268   \int_zero:N \l_tmpb_int
2269
2270   \bool_set_true:N \l_tmpa_bool
2271   \str_map_inline:Nn \l_stex_symdecl_args_str {
2272     \token_case_meaning:NnF ##1 {
2273       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2274       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```



```

2275     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2276     {\tl_to_str:n a} {
2277         \bool_set_false:N \l_tmpa_bool
2278         \int_incr:N \l_tmpb_int
2279     }
2280     {\tl_to_str:n B} {
2281         \bool_set_false:N \l_tmpa_bool
2282         \int_incr:N \l_tmpb_int
2283     }
2284     ){
2285         \msg_error:nnxx{stex}{error/wrongargs}{
2286             \l_stex_current_module_str ?
2287             \l_stex_symdecl_name_str
2288             }{##1}
2289     }
2290 }
2291 \bool_if:NTF \l_tmpa_bool {
2292     % possibly numeric
2293     \str_if_empty:NTF \l_stex_symdecl_args_str {
2294         \prop_put:Nnn \l_tmpa_prop { args } {}
2295         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2296     }{
2297         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2298         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2299         \str_clear:N \l_tmpa_str
2300         \int_step_inline:nn \l_tmpa_int {
2301             \str_put_right:Nn \l_tmpa_str i
2302         }
2303         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2304     }
2305 } {
2306     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2307     \prop_put:Nnx \l_tmpa_prop { arity }
2308     { \str_count:N \l_stex_symdecl_args_str }
2309 }
2310 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2311
2312 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2313     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2314 }{
2315     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2316 }
2317
2318 % semantic macro
2319
2320 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2321     \exp_args:Nx \stex_do_up_to_module:n {
2322         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2323             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2324         }}
2325     }
2326 }
2327
2328 \stex_debug:nn{symbols}{New~symbol:~

```

```

2329 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2330 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2331 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2332 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2333 }
2334
2335 % circular dependencies require this:
2336 \stex_if_do_html:T {
2337   \stex_annotate_invisible:nnn {symdecl} {
2338     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2339   } {
2340     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2341       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2342     }
2343     \stex_annotate_invisible:nnn{args}{\prop_item:Nn \l_tmpa_prop { args }}{ }
2344     \stex_annotate_invisible:nnn{macroname}{#1}{ }
2345     \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2346       \stex_annotate_invisible:nnn{definiens}{ }
2347       {\l_stex_symdecl_definiens_tl$}
2348     }
2349     \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2350       \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{ }
2351     }
2352     \str_if_empty:NF \l_stex_symdecl_reorder_str {
2353       \stex_annotate_invisible:nnn{reorderargs}{\l_stex_symdecl_reorder_str}{ }
2354     }
2355   }
2356 }
2357 \prop_if_exist:cF {
2358   \l_stex_symdecl_
2359   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2360   _prop
2361 } {
2362   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2363   \__stex_symdecl_restore_symbol:nnnnnnn
2364   {\l_stex_symdecl_name_str}
2365   { \prop_item:Nn \l_tmpa_prop {args} }
2366   { \prop_item:Nn \l_tmpa_prop {arity} }
2367   { \prop_item:Nn \l_tmpa_prop {assocs} }
2368   { \prop_item:Nn \l_tmpa_prop {defined} }
2369   {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2370   {\l_stex_current_module_str}
2371 }
2372 }
2373 }
2374 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2375   \prop_clear:N \l_tmpa_prop
2376   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2377   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2378   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2379   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2380   \prop_put:Nnn \l_tmpa_prop { assocs } { #4 }
2381   \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2382   \tl_if_empty:nF{#6}{

```

```

2383 \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2384 }
2385 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2386 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2387 }

```

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

\textsymdecl

```

2388
2389 \keys_define:nn { stex / textsymdecl } {
2390   name      .str_set_x:N = \l__stex_symdecl_name_str ,
2391   type      .tl_set:N   = \l__stex_symdecl_type_tl
2392 }
2393
2394 \cs_new_protected:Nn \_stex_textsymdecl_args:n {
2395   \str_clear:N \l__stex_symdecl_name_str
2396   \tl_clear:N \l__stex_symdecl_type_tl
2397   \keys_set:nn { stex / textsymdecl } { #1 }
2398 }
2399
2400 \NewDocumentCommand \textsymdecl {m O{} m} {
2401   \_stex_textsymdecl_args:n { #2 }
2402   \str_if_empty:NTF \l__stex_symdecl_name_str {
2403     \__stex_symdecl_args:n{name=#1,#2}
2404   }{
2405     \__stex_symdecl_args:n{#2}
2406   }
2407   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2408   \stex_symdecl_do:n{#1-sym}
2409   \stex_execute_in_module:n{
2410     \cs_set_nopar:cpn{#1name}{
2411       \ifvmode\hbox_unpack:N\c_empty_box\fi
2412       \hbox{#3}\xspace
2413     }
2414     \cs_set_nopar:cpn{#1}{
2415       \ifmmode\csname#1-sym\expandafter\endcsname\else
2416       \ifvmode\hbox_unpack:N\c_empty_box\fi
2417       \symref{#1-sym}{\hbox{#3}}\expandafter\xspace
2418       \fi
2419     }
2420   }
2421   \stex_execute_in_module:x{
2422     \__stex_notation_restore_notation:nnnnn
2423     {\l_stex_current_module_str?\tl_if_empty:NTF\l__stex_symdecl_name_str{#1}\l__stex_symdecl
2424     }{0}
2425     {\exp_not:n{\STEXInternalTermMathOMSiiii{\STEXInternalCurrentSymbolStr}{}}{\neginfprec}{
2426       \comp{\hbox{#3}}\STEXInternalSymbolAfterInvokationTL
2427     }}}}
2428   {}
2429 }
2430 \stex_smsmode_do:
2431 }

```

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

`\stex_get_symbol:n`

```
2432 \str_new:N \l_stex_get_symbol_uri_str
2433
2434 \cs_new_protected:Nn \stex_get_symbol:n {
2435   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2436     \tl_set:Nn \l_tmpa_tl { #1 }
2437     \__stex_symdecl_get_symbol_from_cs:
2438   }{
2439     % argument is a string
2440     % is it a command name?
2441     \cs_if_exist:cTF { #1 }{
2442       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2443       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2444       \str_if_empty:NTF \l_tmpa_str {
2445         \exp_args:Nx \cs_if_eq:NNTF {
2446           \tl_head:N \l_tmpa_tl
2447         } \stex_invoke_symbol:n {
2448           \__stex_symdecl_get_symbol_from_cs:
2449         }{
2450           \__stex_symdecl_get_symbol_from_string:n { #1 }
2451         }
2452       } {
2453         \__stex_symdecl_get_symbol_from_string:n { #1 }
2454       }
2455     }{
2456       % argument is not a command name
2457       \__stex_symdecl_get_symbol_from_string:n { #1 }
2458       % \l_stex_all_symbols_seq
2459     }
2460   }
2461   \str_if_eq:eeF {
2462     \prop_item:cn {
2463       l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2464     }{ deprecate }
2465   }{}{
2466     \msg_warning:nxxx{stex}{warning/deprecated}{
2467       Symbol~\l_stex_get_symbol_uri_str
2468     }{
2469       \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2470     }
2471   }
2472 }
2473
2474 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2475   \tl_set:Nn \l_tmpa_tl {
2476     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2477   }
2478   \str_set:Nn \l_tmpa_str { #1 }
2479
2480   %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2481
2482   \str_if_in:NnTF \l_tmpa_str ? {
2483     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2484     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
```

```

2485 \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2486 }{
2487 \str_clear:N \l_tmpb_str
2488 }
2489 \str_if_empty:NTF \l_tmpb_str {
2490 \seq_map_inline:Nn \l_stex_all_modules_seq {
2491 \seq_map_inline:cn{c_stex_module_##1_constants}{
2492 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2493 \seq_map_break:n{\seq_map_break:n{
2494 \tl_set:Nn \l_tmpa_tl {
2495 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2496 }
2497 }}
2498 }
2499 }
2500 }
2501 }{
2502 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2503 \seq_map_inline:Nn \l_stex_all_modules_seq {
2504 \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2505 \seq_map_inline:cn{c_stex_module_##1_constants}{
2506 \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2507 \seq_map_break:n{\seq_map_break:n{
2508 \tl_set:Nn \l_tmpa_tl {
2509 \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2510 }
2511 }}
2512 }
2513 }
2514 }
2515 }
2516 }
2517
2518 \l_tmpa_tl
2519 }
2520
2521 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2522 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2523 { \tl_tail:N \l_tmpa_tl }
2524 \tl_if_single:NTF \l_tmpa_tl {
2525 \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2526 \exp_after:wN \str_set:Nn \exp_after:wN
2527 \l_stex_get_symbol_uri_str \l_tmpa_tl
2528 }{
2529 % TODO
2530 % tail is not a single group
2531 }
2532 }{
2533 % TODO
2534 % tail is not a single group
2535 }
2536 }

```

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

29.2 Notations

```

2537 <@@=stex_notation>

      notation arguments:
2538 \keys_define:nn { stex / notation } {
2539 % lang .tl_set_x:N = \l__stex_notation_lang_str ,
2540 variant .tl_set_x:N = \l__stex_notation_variant_str ,
2541 prec .str_set_x:N = \l__stex_notation_prec_str ,
2542 op .tl_set:N = \l__stex_notation_op_tl ,
2543 primary .bool_set:N = \l__stex_notation_primary_bool ,
2544 primary .default:n = {true} ,
2545 unknown .code:n = \str_set:Nx
2546     \l__stex_notation_variant_str \l_keys_key_str
2547 }
2548
2549 \cs_new_protected:Nn \stex_notation_args:n {
2550 % \str_clear:N \l__stex_notation_lang_str
2551 \str_clear:N \l__stex_notation_variant_str
2552 \str_clear:N \l__stex_notation_prec_str
2553 \tl_clear:N \l__stex_notation_op_tl
2554 \bool_set_false:N \l__stex_notation_primary_bool
2555
2556 \keys_set:nn { stex / notation } { #1 }
2557 }

\notation

2558 \NewDocumentCommand \notation { s m O{}} {
2559 \stex_notation_args:n { #3 }
2560 \tl_clear:N \l_stex_symdecl_definiens_tl
2561 \stex_get_symbol:n { #2 }
2562 \tl_set:Nn \l_stex_notation_after_do_tl {
2563     \__stex_notation_final:
2564     \IfBooleanTF#1{
2565         \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2566     }{}
2567     \stex_smsmode_do:\ignorespacesandpars
2568 }
2569 \stex_notation_do:nnnnn
2570 { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2571 { \prop_item:cn { l_stex_symdecl\l_stex_get_symbol_uri_str_prop } { arity } }
2572 { \l__stex_notation_variant_str }
2573 { \l__stex_notation_prec_str }
2574 }
2575 \stex_deactivate_macro:Nn \notation {module-environments}

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

\stex_notation_do:nnnnn

2576 \seq_new:N \l__stex_notation_precedences_seq
2577 \tl_new:N \l__stex_notation_opprec_tl
2578 \int_new:N \l__stex_notation_currarg_int
2579 \tl_new:N \STEXInternalSymbolAfterInvokationTL
2580
2581 \cs_new_protected:Nn \stex_notation_do:nnnnn {

```

```

2582 \let\STEXInternalCurrentSymbolStr\relax
2583 \seq_clear:N \l__stex_notation_precedences_seq
2584 \tl_clear:N \l__stex_notation_opprec_tl
2585 \str_set:Nx \l__stex_notation_args_str { #1 }
2586 \str_set:Nx \l__stex_notation_arity_str { #2 }
2587 \str_set:Nx \l__stex_notation_suffix_str { #3 }
2588 \str_set:Nx \l__stex_notation_prec_str { #4 }
2589
2590 % precedences
2591 \str_if_empty:NTF \l__stex_notation_prec_str {
2592   \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2593     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2594   }{
2595     \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2596   }
2597 } {
2598   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2599     \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2600     \int_step_inline:nn { \l__stex_notation_arity_str } {
2601       \exp_args:NNo
2602       \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2603     }
2604   }{
2605     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2606     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2607       \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2608       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2609         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2610           \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2611         \seq_map_inline:Nn \l_tmpa_seq {
2612           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2613         }
2614       }
2615     }{
2616       \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2617         \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2618       }{
2619         \tl_set:No \l__stex_notation_opprec_tl { 0 }
2620       }
2621     }
2622   }
2623 }
2624
2625 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2626 \int_step_inline:nn { \l__stex_notation_arity_str } {
2627   \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2628     \exp_args:NNo
2629     \seq_put_right:No \l__stex_notation_precedences_seq {
2630       \l__stex_notation_opprec_tl
2631     }
2632   }
2633 }
2634 \tl_clear:N \l__stex_notation_dummyargs_tl
2635

```

```

2636 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2637   \exp_args:NNe
2638   \cs_set:Npn \l_stex_notation_macrocode_cs {
2639     \STEXInternalTermMathOMSiiii { \STEXInternalCurrentSymbolStr }
2640     { \l__stex_notation_suffix_str }
2641     { \l__stex_notation_opprec_tl }
2642     { \exp_not:n { #5 } }
2643   }
2644   \l_stex_notation_after_do_tl
2645 }{
2646   \str_if_in:NnTF \l__stex_notation_args_str b {
2647     \exp_args:Nne \use:nn
2648     {
2649       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2650       \cs_set:Npn \l__stex_notation_arity_str } { {
2651         \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2652         { \l__stex_notation_suffix_str }
2653         { \l__stex_notation_opprec_tl }
2654         { \exp_not:n { #5 } }
2655       }}
2656   }{
2657     \str_if_in:NnTF \l__stex_notation_args_str B {
2658       \exp_args:Nne \use:nn
2659       {
2660         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2661         \cs_set:Npn \l__stex_notation_arity_str } { {
2662           \STEXInternalTermMathOMBiiii { \STEXInternalCurrentSymbolStr }
2663           { \l__stex_notation_suffix_str }
2664           { \l__stex_notation_opprec_tl }
2665           { \exp_not:n { #5 } }
2666         } }
2667     }{
2668       \exp_args:Nne \use:nn
2669       {
2670         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2671         \cs_set:Npn \l__stex_notation_arity_str } { {
2672           \STEXInternalTermMathOMAiinii { \STEXInternalCurrentSymbolStr }
2673           { \l__stex_notation_suffix_str }
2674           { \l__stex_notation_opprec_tl }
2675           { \exp_not:n { #5 } }
2676         } }
2677     }
2678   }
2679
2680   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2681   \int_zero:N \l__stex_notation_currarg_int
2682   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2683   \__stex_notation_arguments:
2684 }
2685 }

```

(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


```

2686 \cs_new_protected:Nn \__stex_notation_arguments: {
2687   \int_incr:N \l__stex_notation_currarg_int
2688   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2689     \l_stex_notation_after_do_tl
2690   }{
2691     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2692     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2693     \str_if_eq:VnTF \l_tmpa_str a {
2694       \__stex_notation_argument_assoc:nn{a}
2695     }{
2696       \str_if_eq:VnTF \l_tmpa_str B {
2697         \__stex_notation_argument_assoc:nn{B}
2698       }{
2699         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2700         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2701           { \STEXInternalTermMathArgiii
2702             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2703             { \l_tmpb_str }
2704             { ####\int_use:N \l__stex_notation_currarg_int }
2705           }
2706         }
2707         \__stex_notation_arguments:
2708       }
2709     }
2710   }
2711 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2712 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2713
2714   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2715     {\l__stex_notation_arity_str}{
2716       #2
2717     }
2718   \int_zero:N \l_tmpa_int
2719   \tl_clear:N \l_tmpa_tl
2720   \str_map_inline:Nn \l__stex_notation_args_str {
2721     \int_incr:N \l_tmpa_int
2722     \tl_put_right:Nx \l_tmpa_tl {
2723       \str_if_eq:nnTF {##1}{a}{ {} }{
2724         \str_if_eq:nnTF {##1}{B}{ {} }{
2725           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2726         }
2727       }
2728     }
2729   }
2730   \exp_after:wN\exp_after:wN\exp_after:wN \def
2731   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2732   \exp_after:wN\exp_after:wN\exp_after:wN ##
2733   \exp_after:wN\exp_after:wN\exp_after:wN 1
2734   \exp_after:wN\exp_after:wN\exp_after:wN ##
2735   \exp_after:wN\exp_after:wN\exp_after:wN 2

```

```

2736 \exp_after:wN\exp_after:wN\exp_after:wN {
2737   \exp_after:wN \exp_after:wN \exp_after:wN
2738   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2739     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2740   }
2741 }
2742
2743 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2744 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2745   \STEXInternalTermMathAssocArgiiii
2746   { #1\int_use:N \l__stex_notation_currarg_int }
2747   { \l_tmpa_str }
2748   { ####\int_use:N \l__stex_notation_currarg_int }
2749   { \l_tmpa_cs {####1} {####2} }
2750 } }
2751 \__stex_notation_arguments:
2752 }

```

(End definition for __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```

2753 \cs_new_protected:Nn \__stex_notation_restore_notation:nnnnn {
2754   \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}}
2755   \cs_set_nopar:Npn {#3}{#4}
2756   \tl_if_empty:nF {#5}{
2757     \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2758   }
2759   \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2760     \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2761   }
2762 }
2763
2764 \cs_new_protected:Nn \__stex_notation_final: {
2765
2766   \stex_execute_in_module:x {
2767     \__stex_notation_restore_notation:nnnnn
2768     {\l_stex_get_symbol_uri_str}
2769     {\l__stex_notation_suffix_str}
2770     {\l__stex_notation_arity_str}
2771     {
2772       \exp_after:wN \exp_after:wN \exp_after:wN
2773       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2774       { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInt
2775     }
2776     {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2777   }
2778
2779   \stex_debug:nn{symbols}{
2780     Notation~\l__stex_notation_suffix_str
2781     ~for~\l_stex_get_symbol_uri_str^^J
2782     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2783     Argument~precedences:~
2784     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2785     Notation: \cs_meaning:c {

```

```

2786     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2787     \l__stex_notation_suffix_str
2788     _cs
2789 }
2790 }
2791 % HTML annotations
2792 \stex_if_do_html:T {
2793   \stex_annotate_invisible:nnn { notation }
2794   { \l_stex_get_symbol_uri_str } {
2795     \stex_annotate_invisible:nnn { notationfragment }
2796     { \l__stex_notation_suffix_str }{}
2797   \stex_annotate_invisible:nnn { precedence }
2798     { \l__stex_notation_prec_str }{}
2799
2800   \int_zero:N \l_tmpa_int
2801   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2802   \tl_clear:N \l_tmpa_tl
2803   \int_step_inline:nn { \l__stex_notation_arity_str }{
2804     \int_incr:N \l_tmpa_int
2805     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2806     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rem
2807     \str_if_eq:VnTF \l_tmpb_str a {
2808       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2809         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2810         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2811       } }
2812     }{
2813       \str_if_eq:VnTF \l_tmpb_str B {
2814         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2815           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2816           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2817         } }
2818       }{
2819         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2820           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2821         } }
2822       }
2823     }
2824   }
2825   \stex_annotate_invisible:nnn { notationcomp }{}{
2826     \str_set:Nx \STEXInternalCurrentSymbolStr {\l_stex_get_symbol_uri_str }
2827     $ \exp_args:Nno \use:nn { \use:c {
2828       stex_notation_ \STEXInternalCurrentSymbolStr
2829       \c_hash_str \l__stex_notation_suffix_str _cs
2830     } } { \l_tmpa_tl } $
2831   }
2832   \tl_if_empty:NF \l__stex_notation_op_tl {
2833     \stex_annotate_invisible:nnn { notationopcomp }{}{
2834       $\l__stex_notation_op_tl$
2835     }
2836   }
2837 }
2838 }
2839 }

```

(End definition for `_stex_notation_final:`)

`\setnotation`

```

2840 \keys_define:nn { stex / setnotation } {
2841   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2842   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2843   unknown .code:n      = \str_set:Nx
2844     \l__stex_notation_variant_str \l_keys_key_str
2845 }
2846
2847 \cs_new_protected:Nn \stex_setnotation_args:n {
2848   % \str_clear:N \l__stex_notation_lang_str
2849   \str_clear:N \l__stex_notation_variant_str
2850   \keys_set:nn { stex / setnotation } { #1 }
2851 }
2852
2853 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2854   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2855     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2856     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2857   }
2858 }
2859
2860 \cs_new_protected:Nn \stex_setnotation:n {
2861   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2862     { \l__stex_notation_variant_str }{
2863     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2864     \stex_debug:nn {notations}{
2865       Setting~default~notation~
2866       {\l__stex_notation_variant_str }~for~
2867       #1 \\\
2868       \expandafter\meaning\csname
2869       l_stex_symdecl_#1_notations\endcsname
2870     }
2871   }{
2872     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2873   }
2874 }
2875
2876 \NewDocumentCommand \setnotation {m m} {
2877   \stex_get_symbol:n { #1 }
2878   \stex_setnotation_args:n { #2 }
2879   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2880   \stex_smsmode_do:\ignorespacesandpars
2881 }
2882
2883 \cs_new_protected:Nn \stex_copy_notations:nn {
2884   \stex_debug:nn {notations}{
2885     Copying~notations~from~#2~to~#1\\
2886     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2887   }
2888   \tl_clear:N \l_tmpa_tl
2889   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2890     \tl_put_right:Nn \l_tmpa_tl { {##### #1} }

```

```

2891 }
2892 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{\begingroup
2893   \stex_debug:nn{Here}{Here:~##1}
2894   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2895   \edef \l_tmpa_tl {
2896     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2897     \exp_after:wN\exp_after:wN\exp_after:wN {
2898       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2899     }
2900   }
2901
2902   \exp_after:wN \def \exp_after:wN \l_tmpa_tl
2903   \exp_after:wN ####\exp_after:wN 1 \exp_after:wN ####\exp_after:wN 2
2904   \exp_after:wN { \l_tmpa_tl }
2905
2906   \edef \l_tmpa_tl {
2907     \exp_after:wN \exp_not:n \exp_after:wN {
2908       \l_tmpa_tl {##### 1}{##### 2}
2909     }
2910   }
2911
2912   \stex_debug:nn{Here}{Here:~\expandafter\detokenize\expandafter{\l_tmpa_tl}}
2913
2914   \stex_execute_in_module:x {
2915     \__stex_notation_restore_notation:nnnnn
2916     {#1}{##1}
2917     { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2918     { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2919     {
2920       \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2921         \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2922       }
2923     }
2924   }\endgroup
2925 }
2926 }
2927
2928 \NewDocumentCommand \copynotation {m m} {
2929   \stex_get_symbol:n { #1 }
2930   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2931   \stex_get_symbol:n { #2 }
2932   \exp_args:Noo
2933   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2934   \stex_smsmode_do:\ignorespacesandpars
2935 }
2936

```

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

\symdef

```

2937 \keys_define:nn { stex / symdef } {
2938   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2939   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2940   args      .str_set_x:N = \l_stex_symdecl_args_str ,

```

```

2941 type .tl_set:N = \l_stex_symdecl_type_tl ,
2942 def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2943 reorder .str_set_x:N = \l_stex_symdecl_reorder_str ,
2944 op .tl_set:N = \l__stex_notation_op_tl ,
2945 % lang .str_set_x:N = \l__stex_notation_lang_str ,
2946 variant .str_set_x:N = \l__stex_notation_variant_str ,
2947 prec .str_set_x:N = \l__stex_notation_prec_str ,
2948 assoc .choices:nn =
2949 {bin,binl,binr,pre,conj,pwconj}
2950 {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2951 unknown .code:n = \str_set:Nx
2952 \l__stex_notation_variant_str \l_keys_key_str
2953 }
2954
2955 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2956 \str_clear:N \l_stex_symdecl_name_str
2957 \str_clear:N \l_stex_symdecl_args_str
2958 \str_clear:N \l_stex_symdecl_assoctype_str
2959 \str_clear:N \l_stex_symdecl_reorder_str
2960 \bool_set_false:N \l_stex_symdecl_local_bool
2961 \tl_clear:N \l_stex_symdecl_type_tl
2962 \tl_clear:N \l_stex_symdecl_definiens_tl
2963 % \str_clear:N \l__stex_notation_lang_str
2964 \str_clear:N \l__stex_notation_variant_str
2965 \str_clear:N \l__stex_notation_prec_str
2966 \tl_clear:N \l__stex_notation_op_tl
2967
2968 \keys_set:nn { stex / symdef } { #1 }
2969 }
2970
2971 \NewDocumentCommand \symdef { m O{} } {
2972 \__stex_notation_symdef_args:n { #2 }
2973 \bool_set_true:N \l_stex_symdecl_make_macro_bool
2974 \stex_symdecl_do:n { #1 }
2975 \tl_set:Nn \l_stex_notation_after_do_tl {
2976 \__stex_notation_final:
2977 \stex_smsmode_do:\ignorespacesandpars
2978 }
2979 \str_set:Nx \l_stex_get_symbol_uri_str {
2980 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2981 }
2982 \exp_args:Nx \stex_notation_do:nnnnn
2983 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2984 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2985 { \l__stex_notation_variant_str }
2986 { \l__stex_notation_prec_str }
2987 }
2988 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

29.3 Variables

```

2989 <@@=stex_variables>

```

```

2990
2991 \keys_define:nn { stex / vardef } {
2992   name      .str_set_x:N = \l__stex_variables_name_str ,
2993   args      .str_set_x:N = \l__stex_variables_args_str ,
2994   type      .tl_set:N    = \l__stex_variables_type_tl ,
2995   def       .tl_set:N    = \l__stex_variables_def_tl ,
2996   op        .tl_set:N    = \l__stex_variables_op_tl ,
2997   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2998   reorder   .str_set_x:N = \l__stex_variables_reorder_str ,
2999   assoc     .choices:nn  =
3000     {bin,binl,binr,pre,conj,pwconj}
3001     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
3002   bind      .choices:nn  =
3003     {forall,exists}
3004     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3005 }
3006
3007 \cs_new_protected:Nn \__stex_variables_args:n {
3008   \str_clear:N \l__stex_variables_name_str
3009   \str_clear:N \l__stex_variables_args_str
3010   \str_clear:N \l__stex_variables_prec_str
3011   \str_clear:N \l__stex_variables_assoctype_str
3012   \str_clear:N \l__stex_variables_reorder_str
3013   \str_clear:N \l__stex_variables_bind_str
3014   \tl_clear:N \l__stex_variables_type_tl
3015   \tl_clear:N \l__stex_variables_def_tl
3016   \tl_clear:N \l__stex_variables_op_tl
3017
3018   \keys_set:nn { stex / vardef } { #1 }
3019 }
3020
3021 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
3022   \__stex_variables_args:n {#2}
3023   \str_if_empty:NT \l__stex_variables_name_str {
3024     \str_set:Nx \l__stex_variables_name_str { #1 }
3025   }
3026   \prop_clear:N \l_tmpa_prop
3027   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
3028
3029   \int_zero:N \l_tmpb_int
3030   \bool_set_true:N \l_tmpa_bool
3031   \str_map_inline:Nn \l__stex_variables_args_str {
3032     \token_case_meaning:NnF ##1 {
3033       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
3034       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
3035       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
3036       {\tl_to_str:n a} {
3037         \bool_set_false:N \l_tmpa_bool
3038         \int_incr:N \l_tmpb_int
3039       }
3040       {\tl_to_str:n B} {
3041         \bool_set_false:N \l_tmpa_bool
3042         \int_incr:N \l_tmpb_int
3043       }

```

```

3044   }{
3045     \msg_error:nnxx{stex}{error/wrongargs}{
3046       variable~\l__stex_variables_name_str
3047     }{##1}
3048   }
3049 }
3050 \bool_if:NTF \l_tmpa_bool {
3051   % possibly numeric
3052   \str_if_empty:NTF \l__stex_variables_args_str {
3053     \prop_put:Nnn \l_tmpa_prop { args } {}
3054     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
3055   }{
3056     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
3057     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
3058     \str_clear:N \l_tmpa_str
3059     \int_step_inline:nn \l_tmpa_int {
3060       \str_put_right:Nn \l_tmpa_str i
3061     }
3062     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
3063     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3064   }
3065 } {
3066   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
3067   \prop_put:Nnx \l_tmpa_prop { arity }
3068   { \str_count:N \l__stex_variables_args_str }
3069 }
3070 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
3071 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
3072
3073 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
3074
3075 \tl_if_empty:NF \l__stex_variables_op_tl {
3076   \cs_set:cpx {
3077     stex_var_op_notation_\l__stex_variables_name_str _cs
3078   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
3079 }
3080
3081 \tl_set:Nn \l_stex_notation_after_do_tl {
3082   \exp_args:Nne \use:nn {
3083     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
3084     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
3085   } {{
3086     \exp_after:wN \exp_after:wN \exp_after:wN
3087     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
3088     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \STEXInter
3089   }}
3090 \stex_if_do_html:T {
3091   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
3092     \stex_annotate_invisible:nnn { precedence }
3093     { \l__stex_variables_prec_str }{}
3094     \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{-}{\l__
3095     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }{}
3096     \stex_annotate_invisible:nnn{macroname}{#1}{-}
3097     \tl_if_empty:NF \l__stex_variables_def_tl {

```



```

3098     \stex_annotate_invisible:nnn{definiens}{}
3099     {${\l__stex_variables_def_tl$}
3100   }
3101   \str_if_empty:NF \l__stex_variables_assoctype_str {
3102     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{}
3103   }
3104   \str_if_empty:NF \l__stex_variables_reorder_str {
3105     \stex_annotate_invisible:nnn{reorderargs}{\l__stex_variables_reorder_str}{}
3106   }
3107   \int_zero:N \l_tmpa_int
3108   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
3109   \tl_clear:N \l_tmpa_tl
3110   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
3111     \int_incr:N \l_tmpa_int
3112     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
3113     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
3114     \str_if_eq:VnTF \l_tmpb_str a {
3115       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3116         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3117         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3118       } }
3119     }{
3120       \str_if_eq:VnTF \l_tmpb_str B {
3121         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3122           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
3123           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
3124         } }
3125       }{
3126         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3127           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
3128         } }
3129       }
3130     }
3131   }
3132   \stex_annotate_invisible:nnn { notationcomp }{}{
3133     \str_set:Nx \STEXInternalCurrentSymbolStr {var://\l__stex_variables_name_str }
3134     $ \exp_args:Nno \use:nn { \use:c {
3135       stex_var_notation_\l__stex_variables_name_str _cs
3136     } } { \l_tmpa_tl } $
3137   }
3138   \tl_if_empty:NF \l__stex_variables_op_tl {
3139     \stex_annotate_invisible:nnn { notationopcomp }{}{
3140       ${\l__stex_variables_op_tl$
3141     }
3142   }
3143   }
3144   \str_if_empty:NF \l__stex_variables_bind_str {
3145     \stex_annotate_invisible:nnn {bindtype}{\l__stex_variables_bind_str,\l__stex_variab
3146   }
3147   }\ignorespacesandpars
3148 }
3149
3150 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
3151 }

```

```

3152
3153 \cs_new:Nn \_stex_reset:N {
3154   \tl_if_exist:NTF #1 {
3155     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
3156   }{
3157     \let \exp_not:N #1 \exp_not:N \undefined
3158   }
3159 }
3160
3161 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
3162   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
3163   \exp_args:Nnx \use:nn {
3164     % TODO
3165     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
3166       #2
3167     }
3168   }{
3169     \_stex_reset:N \varnot
3170     \_stex_reset:N \vartype
3171     \_stex_reset:N \vardefi
3172   }
3173 }
3174
3175 \NewDocumentCommand \vardef { s } {
3176   \IfBooleanTF#1 {
3177     \__stex_variables_do_complex:nn
3178   }{
3179     \__stex_variables_do_simple:nnn
3180   }
3181 }
3182
3183 \NewDocumentCommand \svar { 0{} m }{
3184   \tl_if_empty:nTF {#1}{
3185     \str_set:Nn \l_tmpa_str { #2 }
3186   }{
3187     \str_set:Nn \l_tmpa_str { #1 }
3188   }
3189   \_stex_term_omv:nn {
3190     var://\l_tmpa_str
3191   }{
3192     \exp_args:Nnx \use:nn {
3193       \def\comp{\_varcomp}
3194       \str_set:Nx \STEXInternalCurrentSymbolStr { var://\l_tmpa_str }
3195       \comp{ #2 }
3196     }{
3197       \_stex_reset:N \comp
3198       \_stex_reset:N \STEXInternalCurrentSymbolStr
3199     }
3200   }
3201 }
3202
3203
3204
3205 \keys_define:nn { stex / varseq } {

```

```

3206 name .str_set_x:N = \l__stex_variables_name_str ,
3207 args .int_set:N = \l__stex_variables_args_int ,
3208 type .tl_set:N = \l__stex_variables_type_tl ,
3209 mid .tl_set:N = \l__stex_variables_mid_tl ,
3210 bind .choices:nn =
3211 {forall,exists}
3212 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3213 }
3214
3215 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3216 \str_clear:N \l__stex_variables_name_str
3217 \int_set:Nn \l__stex_variables_args_int 1
3218 \tl_clear:N \l__stex_variables_type_tl
3219 \str_clear:N \l__stex_variables_bind_str
3220
3221 \keys_set:nn { stex / varseq } { #1 }
3222 }
3223
3224 \NewDocumentCommand \varseq {m O{} m m m}{
3225 \__stex_variables_seq_args:n { #2 }
3226 \str_if_empty:NT \l__stex_variables_name_str {
3227 \str_set:Nx \l__stex_variables_name_str { #1 }
3228 }
3229 \prop_clear:N \l_tmpa_prop
3230 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3231
3232 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3233 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3234 \msg_error:nnxx{stex}{error/seqlength}
3235 {\int_use:N \l__stex_variables_args_int}
3236 {\seq_count:N \l_tmpa_seq}
3237 }
3238 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3239 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3240 \msg_error:nnxx{stex}{error/seqlength}
3241 {\int_use:N \l__stex_variables_args_int}
3242 {\seq_count:N \l_tmpb_seq}
3243 }
3244 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3245 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3246
3247 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3248 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3249
3250 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3251 \int_step_inline:nn \l__stex_variables_args_int {
3252 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3253 }
3254 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3255 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3256 \tl_if_empty:NF \l__stex_variables_mid_tl {
3257 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3258 \tl_put_right:Nn \l_tmpa_tl {\,\ellipses,}
3259 }

```

```

3260 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3261 \int_step_inline:nn \l__stex_variables_args_int {
3262   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3263 }
3264 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3265 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3266
3267
3268 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3269
3270 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3271
3272 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
3273
3274 \int_step_inline:nn \l__stex_variables_args_int {
3275   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3276     \STEXInternalTermMathArgiii{i##1}{0}{\exp_not:n{####}##1}
3277   }}
3278 }
3279
3280 \tl_set:Nx \l_tmpa_tl {
3281   \STEXInternalTermMathOMaiiii { varseq://\l__stex_variables_name_str}{0}{
3282     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3283   }
3284 }
3285
3286 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \STEXInternalSymbolAfterInvokationTL} }
3287
3288 \exp_args:Nno \use:nn {
3289   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3290   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3291
3292   \stex_debug:nn{sequences}{New~Sequence:~
3293     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3294     \prop_to_keyval:N \l_tmpa_prop
3295   }
3296   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3297     \tl_if_empty:NF \l__stex_variables_type_tl {
3298       \stex_annotate:nnn {type}{\l__stex_variables_type_tl$}
3299     }
3300     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3301     \str_if_empty:NF \l__stex_variables_bind_str {
3302       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3303     }
3304     \stex_annotate:nnn{startindex}{\l__stex_variables_startindex}{\l__stex_variables_startindex$}
3305     \stex_annotate:nnn{endindex}{\l__stex_variables_endindex}{\l__stex_variables_endindex$}
3306
3307     \tl_clear:N \l_tmpa_tl
3308     \int_step_inline:nn \l__stex_variables_args_int {
3309       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
3310         \stex_annotate:nnn{argmarker}{##1}{}
3311       } }
3312     }
3313     \stex_annotate_invisible:nnn { notationcomp }{}{

```

```

3314     \str_set:Nx \STEXInternalCurrentSymbolStr {varseq://\l__stex_variables_name_str }
3315     $ \exp_args:Nno \use:nn { \use:c {
3316         stex_varseq_\l__stex_variables_name_str _cs
3317     } } { \l_tmpa_tl } $
3318 }
3319 \stex_annotate_invisible:nnn { notationopcomp }{}{
3320     $ \prop_item:Nn \l_tmpa_prop { notation } $
3321 }
3322
3323 }}
3324
3325 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3326 \ignorespacesandpars
3327 }
3328
3329 </package>

```

Chapter 30

STEX -Terms Implementation

```
3330 <*package>
3331
3332 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3333
3334 <@@=stex_terms>
3335
3336   Warnings and error messages
3337 \msg_new:nnn{stex}{error/nonotation}{
3338   Symbol~#1~invoked,~but~has~no~notation#2!
3339 }
3340 \msg_new:nnn{stex}{error/notationarg}{
3341   Error~in~parsing~notation~#1
3342 }
3343 \msg_new:nnn{stex}{error/noop}{
3344   Symbol~#1~has~no~operator~notation~for~notation~#2
3345 }
3346 \msg_new:nnn{stex}{error/notallowed}{
3347   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3348 }
3349 \msg_new:nnn{stex}{error/doubleargument}{
3350   Argument~#1~of~symbol~#2~already~assigned
3351 }
3352 \msg_new:nnn{stex}{error/overarity}{
3353   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3354 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3354
3355
3356 \bool_new:N \l_stex_allow_semantic_bool
3357 \bool_set_true:N \l_stex_allow_semantic_bool
3358
```

```

3359 \cs_new_protected:Nn \stex_invoke_symbol:n {
3360   \ifvmode\indent\fi
3361   \bool_if:NTF \l_stex_allow_semantic_bool {
3362     \str_if_eq:eeF {
3363       \prop_item:cn {
3364         l_stex_symdecl_#1_prop
3365       }{ deprecate }
3366     }{}{
3367       \msg_warning:nxxx{stex}{warning/deprecated}{
3368         Symbol~#1
3369       }{
3370         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3371       }
3372     }
3373     \if_mode_math:
3374       \exp_after:wN \__stex_terms_invoke_math:n
3375     \else:
3376       \exp_after:wN \__stex_terms_invoke_text:n
3377     \fi: { #1 }
3378   }{
3379     \msg_error:nxxx{stex}{error/notallowed}{#1}{\STEXInternalCurrentSymbolStr}
3380   }
3381 }
3382
3383 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3384   \peek_charcode_remove:NTF ! {
3385     \__stex_terms_invoke_op_custom:nn {#1}
3386   }{
3387     \__stex_terms_invoke_custom:nn {#1}
3388   }
3389 }
3390
3391 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3392   \peek_charcode_remove:NTF ! {
3393     % operator
3394     \peek_charcode_remove:NTF * {
3395       % custom op
3396       \__stex_terms_invoke_op_custom:nn {#1}
3397     }{
3398       % op notation
3399       \peek_charcode:NTF [ {
3400         \__stex_terms_invoke_op_notation:nw {#1}
3401       }{
3402         \__stex_terms_invoke_op_notation:nw {#1}[]
3403       }
3404     }
3405   }{
3406     \peek_charcode_remove:NTF * {
3407       \__stex_terms_invoke_custom:nn {#1}
3408       % custom
3409     }{
3410       % normal
3411       \peek_charcode:NTF [ {
3412         \__stex_terms_invoke_notation:nw {#1}

```

```

3413     }{
3414         \__stex_terms_invoke_notation:nw {#1}[]
3415     }
3416 }
3417 }
3418 }
3419
3420
3421 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3422     \exp_args:Nnx \use:nn {
3423         \def\comp{\_comp}
3424         \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3425         \bool_set_false:N \l_stex_allow_semantic_bool
3426         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3427             \comp{ #2 }
3428         }
3429     }{
3430         \stex_reset:N \comp
3431         \stex_reset:N \STEXInternalCurrentSymbolStr
3432         \bool_set_true:N \l_stex_allow_semantic_bool
3433     }
3434 }
3435
3436 \keys_define:nn { stex / terms } {
3437     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3438     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3439     unknown .code:n      = \str_set:Nx
3440         \l_stex_notation_variant_str \l_keys_key_str
3441 }
3442
3443 \cs_new_protected:Nn \__stex_terms_args:n {
3444     % \str_clear:N \l_stex_notation_lang_str
3445     \str_clear:N \l_stex_notation_variant_str
3446
3447     \keys_set:nn { stex / terms } { #1 }
3448 }
3449
3450 \cs_new_protected:Nn \stex_find_notation:nn {
3451     \__stex_terms_args:n { #2 }
3452     \seq_if_empty:cTF {
3453         l_stex_symdecl_ #1 _notations
3454     } {
3455         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3456     } {
3457         \str_if_empty:NTF \l_stex_notation_variant_str {
3458             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3459         }{
3460             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3461                 \l_stex_notation_variant_str
3462             }{
3463                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3464             }{
3465                 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3466                     ~\l_stex_notation_variant_str

```



```

3467     }
3468   }
3469 }
3470 }
3471 }
3472
3473 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3474   \exp_args:Nnx \use:nn {
3475     \def\comp{\_comp}
3476     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3477     \stex_find_notation:nn { #1 }{ #2 }
3478     \bool_set_false:N \l_stex_allow_semantic_bool
3479     \cs_if_exist:cTF {
3480       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3481     }{
3482       \_stex_term_oms:nnn { #1 }{
3483         #1 \c_hash_str \l_stex_notation_variant_str
3484       }{
3485         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3486       }
3487     }{
3488       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3489         \cs_if_exist:cTF {
3490           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3491         }{
3492           \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3493             \_stex_reset:N \comp
3494             \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3495             \_stex_reset:N \STEXInternalCurrentSymbolStr
3496             \bool_set_true:N \l_stex_allow_semantic_bool
3497           }
3498           \def\comp{\_comp}
3499           \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3500           \bool_set_false:N \l_stex_allow_semantic_bool
3501           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3502         }{
3503           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3504             ~\l_stex_notation_variant_str
3505           }
3506         }
3507       }{
3508         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3509       }
3510     }
3511   }{
3512     \_stex_reset:N \comp
3513     \_stex_reset:N \STEXInternalCurrentSymbolStr
3514     \bool_set_true:N \l_stex_allow_semantic_bool
3515   }
3516 }
3517
3518 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3519   \stex_find_notation:nn { #1 }{ #2 }
3520   \cs_if_exist:cTF {

```

```

3521   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3522 }{
3523   \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
3524     \_stex_reset:N \comp
3525     \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
3526     \_stex_reset:N \STEXInternalCurrentSymbolStr
3527     \bool_set_true:N \l_stex_allow_semantic_bool
3528   }
3529   \def\comp{\_comp}
3530   \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3531   \bool_set_false:N \l_stex_allow_semantic_bool
3532   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3533 }{
3534   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3535     ~\l_stex_notation_variant_str
3536   }
3537 }
3538 }
3539
3540 \prop_new:N \l__stex_terms_custom_args_prop
3541
3542 \cs_new_protected:Nn \__stex_terms_custom_comp:n { \bool_set_false:N \l_stex_allow_semantic_bool
3543
3544 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3545   \exp_args:Nnx \use:nn {
3546     \def\comp{\_stex_terms_custom_comp:n}
3547     \str_set:Nn \STEXInternalCurrentSymbolStr { #1 }
3548     \prop_clear:N \l__stex_terms_custom_args_prop
3549     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3550     \prop_get:cnN {
3551       l_stex_symdecl_#1 _prop
3552     } { args } \l_tmpa_str
3553     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3554     \tl_set:Nn \arg { \_stex_terms_arg: }
3555     \str_if_empty:NTF \l_tmpa_str {
3556       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{\ignorespaces#2}
3557     }{
3558       \str_if_in:NnTF \l_tmpa_str b {
3559         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3560       }{
3561         \str_if_in:NnTF \l_tmpa_str B {
3562           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3563         }{
3564           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{\ignorespaces#2}
3565         }
3566       }
3567     }
3568     % TODO check that all arguments exist
3569   }{
3570     \_stex_reset:N \STEXInternalCurrentSymbolStr
3571     \_stex_reset:N \arg
3572     \_stex_reset:N \comp
3573     \_stex_reset:N \l__stex_terms_custom_args_prop
3574     %\bool_set_true:N \l_stex_allow_semantic_bool

```

```

3575 }
3576 }
3577
3578 \NewDocumentCommand \_stex_terms_arg: { s O{} m}{
3579   \tl_if_empty:nTF {#2}{
3580     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3581     \bool_set_true:N \l_tmpa_bool
3582     \bool_do_while:Nn \l_tmpa_bool {
3583       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3584         \int_incr:N \l_tmpa_int
3585       }{
3586         \bool_set_false:N \l_tmpa_bool
3587       }
3588     }
3589   }{
3590     \int_set:Nn \l_tmpa_int { #2 }
3591   }
3592   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3593   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3594     \msg_error:nnxxx{stex}{error/overarity}
3595     {\int_use:N \l_tmpa_int}
3596     {\STEXInternalCurrentSymbolStr}
3597     {\str_count:N \l_tmpa_str}
3598   }
3599   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3600   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3601     \bool_lazy_any:nF {
3602       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3603       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3604     }{
3605       \msg_error:nnxx{stex}{error/doubleargument}
3606       {\int_use:N \l_tmpa_int}
3607       {\STEXInternalCurrentSymbolStr}
3608     }
3609   }
3610   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {\ign
3611   \bool_set_true:N \l_stex_allow_semantic_bool
3612   \IfBooleanTF#1{
3613     \stex_annotate_invisible:n { %TODO
3614       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3615     }
3616   }{ %TODO
3617     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{\ignorespaces#3}
3618   }
3619   \bool_set_false:N \l_stex_allow_semantic_bool
3620 }
3621
3622
3623 \cs_new_protected:Nn \_stex_term_arg:nn {
3624   \bool_set_true:N \l_stex_allow_semantic_bool
3625   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3626   \bool_set_false:N \l_stex_allow_semantic_bool
3627 }
3628

```

```

3629 \cs_new_protected:Npn \STEXInternalTermMathArgiii #1#2#3 {
3630   \exp_args:Nnx \use:nn
3631     { \int_set:Nn \l__stex_terms_downprec { #2 }
3632       \stex_term_arg:nn { #1 }{ #3 }
3633     }
3634     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3635   }

```

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

`\STEXInternalTermMathAssocArgiii`

```

3636 \cs_new_protected:Npn \STEXInternalTermMathAssocArgiii #1#2#3#4 {
3637   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3638   \tl_set:Nn \l_tmpb_tl {\STEXInternalTermMathArgiii{#1}{#2}}
3639   \tl_if_empty:nTF { #3 }{
3640     \STEXInternalTermMathArgiii{#1}{#2}{}
3641   }{
3642     \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3643       \expandafter\if\expandafter\relax\noexpand#3
3644         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3645       \else
3646         \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3647       \fi
3648       \l_tmpa_tl
3649     }{
3650       \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3651     }
3652   }
3653 }
3654
3655 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3656   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3657   \str_if_empty:NTF \l_tmpa_str {
3658     \exp_args:Nx \cs_if_eq:NNTF {
3659       \tl_head:N #1
3660     } \stex_invoke_sequence:n {
3661       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3662       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3663       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq_\l_tmpa_str _prop}{notation}}
3664       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3665       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3666         \exp_not:n{\exp_args:Nnx \use:nn} {
3667           \exp_not:n {
3668             \def\comp{\_varcomp}
3669             \str_set:Nn \STEXInternalCurrentSymbolStr
3670               { \varseq:/{\l_tmpa_str}
3671             \exp_not:n{ ##1 }
3672           }{
3673             \exp_not:n {
3674               \stex_reset:N \comp
3675               \stex_reset:N \STEXInternalCurrentSymbolStr
3676             }
3677           }
3678         }}}

```

```

3679 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3680 \seq_reverse:N \l_tmpa_seq
3681 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3682 \seq_map_inline:Nn \l_tmpa_seq {
3683   \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3684     \exp_args:Nno
3685     \l_tmpa_cs { ##1 } \l_tmpa_tl
3686   }
3687 }
3688 \tl_set:Nx \l_tmpa_tl {
3689   \stex_term_omv:nn {varseq://\l_tmpa_str}{
3690     \exp_args:No \exp_not:n \l_tmpa_tl
3691   }
3692 }
3693 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3694 }{
3695   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3696 }
3697 } {
3698   \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3699 }
3700 }
3701 }
3702
3703 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3704   \clist_set:Nn \l_tmpa_clist{ #2 }
3705   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3706     \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3707   }{
3708     \clist_reverse:N \l_tmpa_clist
3709     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3710     \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3711       \exp_args:No \exp_not:n \l_tmpa_tl
3712     }}
3713     \clist_map_inline:Nn \l_tmpa_clist {
3714       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3715         \exp_args:Nno
3716         \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3717       }
3718     }
3719   }
3720   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3721 }

```

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

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3722 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3723 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}

```

```

3724 \int_new:N \l__stex_terms_downprec
3725 \int_set_eq:NN \l__stex_terms_downprec \infpref

```

(End definition for `\infpref`, `\neginfpref`, 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

```

```

3726 \tl_set:Nn \l__stex_terms_left_bracket_str (
3727 \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

```

3728 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3729   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3730     \bool_set_false:N \l__stex_terms_brackets_done_bool
3731     #2
3732   } {
3733     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3734       \bool_if:NTF \l__stex_inarray_bool { #2 } {
3735         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3736         \dobrackets { #2 }
3737       }
3738     }{ #2 }
3739   }
3740 }

```

(End definition for `__stex_terms_maybe_brackets:nn`.)

\dobrackets

```

3741 \bool_new:N \l__stex_terms_brackets_done_bool
3742 %\RequirePackage{scalerel}
3743 \cs_new_protected:Npn \dobrackets #1 {
3744   %\ThisStyle{\if D\m@switch
3745   %   \exp_args:Nnx \use:nn
3746   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3747   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3748   %   \else
3749   %   \exp_args:Nnx \use:nn
3750   %   {
3751     \bool_set_true:N \l__stex_terms_brackets_done_bool
3752     \int_set:Nn \l__stex_terms_downprec \infpref
3753     \l__stex_terms_left_bracket_str
3754     #1
3755   }
3756   {
3757     \bool_set_false:N \l__stex_terms_brackets_done_bool
3758     \l__stex_terms_right_bracket_str
3759     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3760   }
3761   %\fi}
3762 }

```

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

\withbrackets

```
3763 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3764   \exp_args:Nnx \use:nn
3765   {
3766     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3767     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3768     #3
3769   }
3770   {
3771     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3772     {\l__stex_terms_left_bracket_str}
3773     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3774     {\l__stex_terms_right_bracket_str}
3775   }
3776 }
```

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

\STEXinvisible

```
3777 \cs_new_protected:Npn \STEXinvisible #1 {
3778   \stex_annotate_invisible:n { #1 }
3779 }
```

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

OMDoc terms:

\STEXInternalTermMathOMSiiii

```
3780 \cs_new_protected:Nn \_stex_term_oms:nnn {
3781   \stex_annotate:nnn{ OMID }{ #2 }{
3782     #3
3783   }
3784 }
3785
3786 \cs_new_protected:Npn \STEXInternalTermMathOMSiiii #1#2#3#4 {
3787   \_stex_terms_maybe_brackets:nn { #3 }{
3788     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3789   }
3790 }
```

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

_stex_term_math_omv:nn

```
3791 \cs_new_protected:Nn \_stex_term_omv:nn {
3792   \stex_annotate:nnn{ OMV }{ #1 }{
3793     #2
3794   }
3795 }
```

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

\STEXInternalTermMathOMAi

```
3796 \cs_new_protected:Nn \_stex_term_oma:nnn {
3797   \stex_annotate:nnn{ OMA }{ #2 }{
3798     #3
3799   }
```

```

3800 }
3801
3802 \cs_new_protected:Npn \STEXInternalTermMathOMAiiiii #1#2#3#4 {
3803   \__stex_terms_maybe_brackets:nn { #3 }{
3804     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3805   }
3806 }

```

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

\STEXInternalTermMathOMBiiiii

```

3807 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3808   \stex_annotate:nnn{ OMBIND }{ #2 }{
3809     #3
3810   }
3811 }
3812
3813 \cs_new_protected:Npn \STEXInternalTermMathOMBiiiii #1#2#3#4 {
3814   \__stex_terms_maybe_brackets:nn { #3 }{
3815     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3816   }
3817 }

```

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

\symref
\symname

```

3818 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3819
3820 \keys_define:nn { stex / symname } {
3821   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3822   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3823   root     .tl_set_x:N      = \l__stex_terms_root_tl
3824 }
3825
3826 \cs_new_protected:Nn \stex_symname_args:n {
3827   \tl_clear:N \l__stex_terms_post_tl
3828   \tl_clear:N \l__stex_terms_pre_tl
3829   \tl_clear:N \l__stex_terms_root_str
3830   \keys_set:nn { stex / symname } { #1 }
3831 }
3832
3833 \NewDocumentCommand \symref { m m }{
3834   \let\compemph_uri_prev:\compemph@uri
3835   \let\compemph@uri\symrefemph@uri
3836   \STEXsymbol{#1}!\{ #2 }
3837   \let\compemph@uri\compemph_uri_prev:
3838 }
3839
3840 \NewDocumentCommand \synonym { O{} m m }{
3841   \stex_symname_args:n { #1 }
3842   \let\compemph_uri_prev:\compemph@uri
3843   \let\compemph@uri\symrefemph@uri
3844   % TODO
3845   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3846   \let\compemph@uri\compemph_uri_prev:

```



```

3847 }
3848
3849 \NewDocumentCommand \symname { 0{} m }{
3850   \stex_symname_args:n { #1 }
3851   \stex_get_symbol:n { #2 }
3852   \str_set:Nx \l_tmpa_str {
3853     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3854   }
3855   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3856
3857   \let\compemph_uri_prev:\compemph@uri
3858   \let\compemph@uri\symrefemph@uri
3859   \exp_args:NNx \use:nn
3860   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3861     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3862   } }
3863   \let\compemph@uri\compemph_uri_prev:
3864 }
3865
3866 \NewDocumentCommand \Symname { 0{} m }{
3867   \stex_symname_args:n { #1 }
3868   \stex_get_symbol:n { #2 }
3869   \str_set:Nx \l_tmpa_str {
3870     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3871   }
3872   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3873   \let\compemph_uri_prev:\compemph@uri
3874   \let\compemph@uri\symrefemph@uri
3875   \exp_args:NNx \use:nn
3876   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmmode*\fi{
3877     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3878     \l__stex_terms_post_tl
3879   } }
3880   \let\compemph@uri\compemph_uri_prev:
3881 }

```

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

30.3 Notation Components

```

3882 <@@=stex_notationcomps>

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

3883 \cs_new_protected:Npn \_comp #1 {
3884   \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3885     \stex_html_backend:TF {
3886       \stex_annotate:nnn { comp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3887     }{
3888       \exp_args:Nnx \compemph@uri { #1 } { \STEXInternalCurrentSymbolStr }
3889     }
3890   }
3891 }
3892
3893 \cs_new_protected:Npn \_varcomp #1 {

```

```

3894 \str_if_empty:NF \STEXInternalCurrentSymbolStr {
3895   \stex_html_backend:TF {
3896     \stex_annotate:nnn { varcomp }{ \STEXInternalCurrentSymbolStr }{ #1 }
3897   }{
3898     \exp_args:Nnx \varemp@uri { #1 } { \STEXInternalCurrentSymbolStr }
3899   }
3900 }
3901 }
3902
3903 \def\comp{\_comp}
3904
3905 \cs_new_protected:Npn \compemph@uri #1 #2 {
3906   \compemph{ #1 }
3907 }
3908
3909
3910 \cs_new_protected:Npn \compemph #1 {
3911   #1
3912 }
3913
3914 \cs_new_protected:Npn \defemph@uri #1 #2 {
3915   \defemph{#1}
3916 }
3917
3918 \cs_new_protected:Npn \defemph #1 {
3919   \textbf{#1}
3920 }
3921
3922 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3923   \symrefemph{#1}
3924 }
3925
3926 \cs_new_protected:Npn \symrefemph #1 {
3927   \emph{#1}
3928 }
3929
3930 \cs_new_protected:Npn \varemp@uri #1 #2 {
3931   \varemp{#1}
3932 }
3933
3934 \cs_new_protected:Npn \varemp #1 {
3935   #1
3936 }

```

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

\ellipses

```

3937 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3938 \bool_new:N \l_stex_inparray_bool
\parrayline 3939 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3940 \NewDocumentCommand \parray { m m } {
\parraycell

```

```

3941 \begingroup
3942 \bool_set_true:N \l_stex_inarray_bool
3943 \begin{array}{#1}
3944 #2
3945 \end{array}
3946 \endgroup
3947 }
3948
3949 \NewDocumentCommand \prmatrix { m } {
3950 \begingroup
3951 \bool_set_true:N \l_stex_inarray_bool
3952 \begin{matrix}
3953 #1
3954 \end{matrix}
3955 \endgroup
3956 }
3957
3958 \def \maybepline {
3959 \bool_if:NT \l_stex_inarray_bool {\hline}
3960 }
3961
3962 \def \parrayline #1 #2 {
3963 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3964 }
3965
3966 \def \pmrow #1 { \parrayline{}{ #1 } }
3967
3968 \def \parraylineh #1 #2 {
3969 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3970 }
3971
3972 \def \parraycell #1 {
3973 #1 \bool_if:NT \l_stex_inarray_bool {&}
3974 }

```

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

30.4 Variables

```

3975 <@@=stex_variables>

```

`\stex_invoke_variable:n` Invokes a variable

```

3976 \cs_new_protected:Nn \stex_invoke_variable:n {
3977 \if_mode_math:
3978 \exp_after:wN \__stex_variables_invoke_math:n
3979 \else:
3980 \exp_after:wN \__stex_variables_invoke_text:n
3981 \fi: {#1}
3982 }
3983
3984 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3985 \peek_charcode_remove:NTF ! {
3986 \__stex_variables_invoke_op_custom:nn {#1}
3987 }{

```

```

3988     \_stex_variables_invoke_custom:nn {#1}
3989   }
3990 }
3991
3992
3993 \cs_new_protected:Nn \_stex_variables_invoke_math:n {
3994   \peek_charcode_remove:NTF ! {
3995     \peek_charcode_remove:NTF ! {
3996       \peek_charcode:NTF [ {
3997         % TODO throw error
3998       }{
3999         \_stex_variables_invoke_op_custom:nn
4000       }
4001     }{
4002       \_stex_variables_invoke_op:n { #1 }
4003     }
4004   }{
4005     \peek_charcode_remove:NTF * {
4006       \_stex_variables_invoke_custom:nn { #1 }
4007     }{
4008       \_stex_variables_invoke_math_ii:n { #1 }
4009     }
4010   }
4011 }
4012
4013 \cs_new_protected:Nn \_stex_variables_invoke_op_custom:nn {
4014   \exp_args:Nnx \use:nn {
4015     \def\comp{\_varcomp}
4016     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4017     \bool_set_false:N \l_stex_allow_semantic_bool
4018     \_stex_term_omv:nn {var://#1}{
4019       \comp{ #2 }
4020     }
4021   }{
4022     \_stex_reset:N \comp
4023     \_stex_reset:N \STEXInternalCurrentSymbolStr
4024     \bool_set_true:N \l_stex_allow_semantic_bool
4025   }
4026 }
4027
4028 \cs_new_protected:Nn \_stex_variables_invoke_op:n {
4029   \cs_if_exist:cTF {
4030     stex_var_op_notation_ #1 _cs
4031   }{
4032     \exp_args:Nnx \use:nn {
4033       \def\comp{\_varcomp}
4034       \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4035       \_stex_term_omv:nn { var://#1 }{
4036         \use:c{stex_var_op_notation_ #1 _cs }
4037       }
4038     }{
4039       \_stex_reset:N \comp
4040       \_stex_reset:N \STEXInternalCurrentSymbolStr
4041     }

```

```

4042 }{
4043   \int_compare:nNnTF {\prop_item:cn {l_stex_variable_#1_prop}{arity}} = 0{
4044     \__stex_variables_invoke_math_ii:n {#1}
4045   }{
4046     \msg_error:nnxx{stex}{error/noop}{variable~#1}{}
4047   }
4048 }
4049 }
4050
4051 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
4052   \cs_if_exist:cTF {
4053     stex_var_notation_#1_cs
4054   }{
4055     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4056       \_stex_reset:N \comp
4057       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4058       \_stex_reset:N \STEXInternalCurrentSymbolStr
4059       \bool_set_true:N \l_stex_allow_semantic_bool
4060     }
4061     \def\comp{\_varcomp}
4062     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4063     \bool_set_false:N \l_stex_allow_semantic_bool
4064     \use:c{stex_var_notation_#1_cs}
4065   }{
4066     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
4067   }
4068 }
4069
4070 \cs_new_protected:Nn \__stex_variables_invoke_custom:nn {
4071   \exp_args:Nnx \use:nn {
4072     \def\comp{\_varcomp}
4073     \str_set:Nn \STEXInternalCurrentSymbolStr { var://#1 }
4074     \prop_clear:N \l__stex_terms_custom_args_prop
4075     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
4076     \prop_get:cnN {
4077       l_stex_variable_#1_prop
4078     }{ args } \l_tmpa_str
4079     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
4080     \tl_set:Nn \arg { \__stex_terms_arg: }
4081     \str_if_empty:NTF \l_tmpa_str {
4082       \_stex_term_omv:nn {var://#1}{\ignorespaces#2}
4083     }{
4084       \str_if_in:NnTF \l_tmpa_str b {
4085         \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4086       }{
4087         \str_if_in:NnTF \l_tmpa_str B {
4088           \_stex_term_ombind:nnn {var://#1}{\ignorespaces#2}
4089         }{
4090           \_stex_term_oma:nnn {var://#1}{\ignorespaces#2}
4091         }
4092       }
4093     }
4094     % TODO check that all arguments exist
4095   }{

```

```

4096 \_stex_reset:N \STEXInternalCurrentSymbolStr
4097 \_stex_reset:N \arg
4098 \_stex_reset:N \comp
4099 \_stex_reset:N \l__stex_terms_custom_args_prop
4100 %\bool_set_true:N \l_stex_allow_semantic_bool
4101 }
4102 }

```

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

30.5 Sequences

```

4103 <@@=stex_sequences>
4104
4105 \cs_new_protected:Nn \stex_invoke_sequence:n {
4106   \peek_charcode_remove:NTF ! {
4107     \_stex_term_omv:nn {varseq://#1}{
4108       \exp_args:Nnx \use:nn {
4109         \def\comp{\_varcomp}
4110         \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4111         \prop_item:cn{stex_varseq_#1_prop}{notation}
4112       }{
4113         \_stex_reset:N \comp
4114         \_stex_reset:N \STEXInternalCurrentSymbolStr
4115       }
4116     }
4117   }{
4118     \bool_set_false:N \l_stex_allow_semantic_bool
4119     \def\comp{\_varcomp}
4120     \str_set:Nn \STEXInternalCurrentSymbolStr {varseq://#1}
4121     \tl_set:Nx \STEXInternalSymbolAfterInvokationTL {
4122       \_stex_reset:N \comp
4123       \_stex_reset:N \STEXInternalSymbolAfterInvokationTL
4124       \_stex_reset:N \STEXInternalCurrentSymbolStr
4125       \bool_set_true:N \l_stex_allow_semantic_bool
4126     }
4127     \use:c { stex_varseq_#1_cs }
4128   }
4129 }
4130 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
4131 ⟨*package⟩
4132
4133 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4134
      Warnings and error messages
4135 \msg_new:nnn{stex}{error/copymodule/notallowed}{
4136   Symbol~#1~can~not~be~assigned~in~copymodule~#2
4137 }
4138 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
4139   Symbol~#1~not~assigned~in~interpretmodule~#2
4140 }
4141
4142 \msg_new:nnn{stex}{error/unknownstructure}{
4143   No~structure~#1~found!
4144 }
4145
4146 \msg_new:nnn{stex}{error/unknownfield}{
4147   No~field~#1~in~instance~#2~found!\#3
4148 }
4149
4150 \msg_new:nnn{stex}{error/keyval}{
4151   Invalid~key=value~pair~#1
4152 }
4153 \msg_new:nnn{stex}{error/instantiate/missing}{
4154   Assignments~missing~in~instantiate:~#1
4155 }
4156 \msg_new:nnn{stex}{error/incompatible}{
4157   Incompatible~signature:~#1~(#2)~and~#3~(#4)
4158 }
4159
```

31.1 Imports with modification

```

4160 <@@=stex_copymodule>
4161 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
4162   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4163     \tl_set:Nn \l_tmpa_tl { #1 }
4164     \__stex_copymodule_get_symbol_from_cs:
4165   }{
4166     % argument is a string
4167     % is it a command name?
4168     \cs_if_exist:cTF { #1 }{
4169       \cs_set_eq:Nc \l_tmpa_tl { #1 }
4170       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4171       \str_if_empty:NTF \l_tmpa_str {
4172         \exp_args:Nx \cs_if_eq:NNTF {
4173           \tl_head:N \l_tmpa_tl
4174         } \stex_invoke_symbol:n {
4175           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
4176         }{
4177           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4178         }
4179       } {
4180         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4181       }
4182     }{
4183       % argument is not a command name
4184       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
4185       % \l_stex_all_symbols_seq
4186     }
4187   }
4188 }
4189
4190 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
4191   \str_set:Nn \l_tmpa_str { #1 }
4192   \bool_set_false:N \l_tmpa_bool
4193   \bool_if:NF \l_tmpa_bool {
4194     \tl_set:Nn \l_tmpa_tl {
4195       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
4196     }
4197     \str_set:Nn \l_tmpa_str { #1 }
4198     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4199     \seq_map_inline:Nn #2 {
4200       \str_set:Nn \l_tmpb_str { ##1 }
4201       \str_if_eq:eeT { \l_tmpa_str } {
4202         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4203       } {
4204         \seq_map_break:n {
4205           \tl_set:Nn \l_tmpa_tl {
4206             \str_set:Nn \l_stex_get_symbol_uri_str {
4207               ##1
4208             }
4209           }
4210         }
4211       }

```



```

4212     }
4213     \l_tmpa_tl
4214 }
4215 }
4216
4217 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
4218   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4219     { \tl_tail:N \l_tmpa_tl }
4220   \tl_if_single:NTF \l_tmpa_tl {
4221     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
4222       \exp_after:wN \str_set:Nn \exp_after:wN
4223         \l_stex_get_symbol_uri_str \l_tmpa_tl
4224       \__stex_copymodule_get_symbol_check:n { #1 }
4225     }{
4226       % TODO
4227       % tail is not a single group
4228     }
4229   }{
4230     % TODO
4231     % tail is not a single group
4232   }
4233 }
4234
4235 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4236   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4237     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4238       :~\seq_use:Nn #1 {,~}
4239     }
4240   }
4241 }
4242
4243 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4244   % import module
4245   \stex_import_module_uri:nn { #1 } { #2 }
4246   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4247   \stex_import_require_module:nnnn
4248     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4249     { \l_stex_import_path_str } { \l_stex_import_name_str }
4250
4251   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4252   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4253
4254   % fields
4255   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4256   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4257     \seq_map_inline:cn {c_stex_module_##1_constants}{
4258       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4259         ##1 ? #####1
4260       }
4261     }
4262   }
4263
4264   % setup prop
4265   \seq_clear:N \l_tmpa_seq

```

```

4266 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4267   name      = \l_stex_current_copymodule_name_str ,
4268   module    = \l_stex_current_module_str ,
4269   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4270   includes  = \l_tmpa_seq %,
4271 % fields    = \l_tmpa_seq
4272 }
4273 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4274   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4275 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {
4276 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {
4277
4278 \stex_if_do_html:T {
4279   \begin{stex_annotate_env} {#4} {
4280     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4281   }
4282   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{
4283 }
4284 }
4285
4286 \cs_new_protected:Nn \stex_copymodule_end:n {
4287   % apply to every field
4288   \def \l_tmpa_cs ##1 ##2 {#1}
4289
4290   \tl_clear:N \__stex_copymodule_module_tl
4291   \tl_clear:N \__stex_copymodule_exec_tl
4292
4293   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4294   \seq_clear:N \__stex_copymodule_fields_seq
4295
4296   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4297     \seq_map_inline:cn {c_stex_module_##1_constants}{
4298
4299       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4300       \l_tmpa_cs{##1}{####1}
4301
4302       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4303         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4304         \stex_if_do_html:T {
4305           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4306             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4307           }
4308         }
4309       }{
4310         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4311       }
4312
4313       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4314       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4315       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4316
4317       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4318         \stex_if_do_html:T {
4319           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4320         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4321     }
4322 }
4323 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4324 }
4325
4326 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4327 \tl_put_right:Nx \__stex_copymodule_module_tl {
4328     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4329     \prop_set_from_keyval:cn {
4330         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4331     }{
4332         \prop_to_keyval:N \l_tmpa_prop
4333     }
4334 }
4335
4336 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4337     \stex_if_do_html:T {
4338         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4339             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4340         }
4341     }
4342     \tl_put_right:Nx \__stex_copymodule_module_tl {
4343         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4344             \stex_invoke_symbol:n {
4345                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4346             }
4347         }
4348     }
4349 }
4350
4351 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4352
4353 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4354     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4355 }
4356
4357 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4358     \stex_if_do_html:TF{
4359         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4360     }{
4361         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4362     }
4363 }
4364 }
4365 }
4366
4367
4368 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4369 \tl_put_left:Nx \__stex_copymodule_module_tl {
4370     \prop_set_from_keyval:cn {
4371         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4372     }{
4373         \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4374     }
4375 }
4376
4377 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4378   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4379 }
4380
4381 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4382 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4383 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4384
4385 \__stex_copymodule_exec_tl
4386 \stex_if_do_html:T {
4387   \end{stex_annotate_env}
4388 }
4389 }
4390
4391 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4392   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4393   \stex_deactivate_macro:Nn \symdecl {module~environments}
4394   \stex_deactivate_macro:Nn \symdef {module~environments}
4395   \stex_deactivate_macro:Nn \notation {module~environments}
4396   \stex_reactivate_macro:N \assign
4397   \stex_reactivate_macro:N \renamedekl
4398   \stex_reactivate_macro:N \donotcopy
4399   \stex_smsmode_do:
4400 }{
4401   \stex_copymodule_end:n {}
4402 }
4403
4404 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4405   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4406   \stex_deactivate_macro:Nn \symdecl {module~environments}
4407   \stex_deactivate_macro:Nn \symdef {module~environments}
4408   \stex_deactivate_macro:Nn \notation {module~environments}
4409   \stex_reactivate_macro:N \assign
4410   \stex_reactivate_macro:N \renamedekl
4411   \stex_reactivate_macro:N \donotcopy
4412   \stex_smsmode_do:
4413 }{
4414   \stex_copymodule_end:n {
4415     \tl_if_exist:cF {
4416       l__stex_copymodule_copymodule_##1?##2_def_tl
4417     }{
4418       \str_if_eq:eeF {
4419         \prop_item:cn{
4420           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4421       }{ true }{
4422         \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4423           ##1?##2
4424         }{\l_stex_current_copymodule_name_str}
4425       }
4426     }
4427   }

```

```

4428 }
4429
4430 \iffalse \begin{stex_annotate_env} \fi
4431 \NewDocumentEnvironment {realization} { 0 } { m } {
4432   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4433   \stex_deactivate_macro:Nn \symdecl {module~environments}
4434   \stex_deactivate_macro:Nn \symdef {module~environments}
4435   \stex_deactivate_macro:Nn \notation {module~environments}
4436   \stex_reactivate_macro:N \donotcopy
4437   \stex_reactivate_macro:N \assign
4438   \stex_smsmode_do:
4439 } {
4440   \stex_import_module_uri:nn { #1 } { #2 }
4441   \tl_clear:N \__stex_copymodule_exec_tl
4442   \tl_set:Nx \__stex_copymodule_module_tl {
4443     \stex_import_require_module:nnnn
4444     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4445     { \l_stex_import_path_str } { \l_stex_import_name_str }
4446   }
4447
4448   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4449     \seq_map_inline:cn {c_stex_module_##1_constants}{
4450       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4451       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4452         \stex_if_do_html:T {
4453           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4454             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4455               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4456             }
4457           }
4458         }
4459         \tl_put_right:Nx \__stex_copymodule_module_tl {
4460           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4461         }
4462       }
4463     }
4464
4465     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4466
4467     \__stex_copymodule_exec_tl
4468     \stex_if_do_html:T {\end{stex_annotate_env}}
4469   }
4470
4471   \NewDocumentCommand \donotcopy { m } {
4472     \str_clear:N \l_stex_import_name_str
4473     \str_set:Nn \l_tmpa_str { #1 }
4474     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4475     \seq_map_inline:Nn \l_stex_all_modules_seq {
4476       \str_set:Nn \l_tmpb_str { ##1 }
4477       \str_if_eq:eeT { \l_tmpa_str } {
4478         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4479       } {
4480         \seq_map_break:n {
4481           \stex_if_do_html:T {

```

```

4482         \stex_if_smsmode:F {
4483             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4484                 \stex_annotate:nnn{domain}{##1}{}}
4485         }
4486     }
4487 }
4488 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4489 }
4490 }
4491 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4492     \str_set:Nn \l_tmpb_str { #####1 }
4493     \str_if_eq:eeT { \l_tmpa_str } {
4494         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4495     } {
4496         \seq_map_break:n {\seq_map_break:n {
4497             \stex_if_do_html:T {
4498                 \stex_if_smsmode:F {
4499                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4500                         \stex_annotate:nnn{domain}{
4501                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4502                         }}
4503                 }
4504             }
4505         }
4506         \str_set:Nx \l_stex_import_name_str {
4507             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4508         }
4509     }}
4510 }
4511 }
4512 }
4513 \str_if_empty:NTF \l_stex_import_name_str {
4514     % TODO throw error
4515 }{
4516     \stex_collect_imports:n {\l_stex_import_name_str }
4517     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4518         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4519         \seq_map_inline:cn {c_stex_module_###1_constants}{
4520             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4521             \bool_lazy_any:nT {
4522                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_name_str}}
4523                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_macroname_str}}
4524                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?#####1_def_tl}}
4525             }{
4526                 % TODO throw error
4527             }
4528         }
4529     }
4530     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4531     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4532     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4533 }
4534 \stex_smsmode_do:
4535 }

```

```

4536 \NewDocumentCommand \assign { m m }{
4537   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4538   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4539   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4540   \stex_smsmode_do:
4541 }
4542
4543
4544 \keys_define:nn { stex / renamedecl } {
4545   name .str_set_x:N = \l_stex_renamedecl_name_str
4546 }
4547 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4548   \str_clear:N \l_stex_renamedecl_name_str
4549   \keys_set:nn { stex / renamedecl } { #1 }
4550 }
4551
4552 \NewDocumentCommand \renamedecl { O{} m m }{
4553   \__stex_copymodule_renamedecl_args:n { #1 }
4554   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4555   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4556   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4557   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4558     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4559       \l_stex_get_symbol_uri_str
4560     } }
4561   } {
4562     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4563       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4564       \prop_set_eq:cc {l_stex_symdecl_
4565         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4566         _prop
4567       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4568       \seq_set_eq:cc {l_stex_symdecl_
4569         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4570         _notations
4571       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4572       \prop_put:cnx {l_stex_symdecl_
4573         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4574         _prop
4575       }{ name }{ \l_stex_renamedecl_name_str }
4576       \prop_put:cnx {l_stex_symdecl_
4577         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4578         _prop
4579       }{ module }{ \l_stex_current_module_str }
4580       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4581         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4582       }
4583       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4584         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4585       } }
4586     }
4587   \stex_smsmode_do:
4588 }
4589

```

```

4590 \stex_deactivate_macro:Nn \assign {copymodules}
4591 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4592 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4593
4594

```

31.2 The feature environment

structural@feature

```

4595 <@@=stex_features>
4596
4597 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4598   \stex_if_in_module:F {
4599     \msg_set:nnn{stex}{error/nomodule}{
4600       Structural~Feature~has~to~occur~in~a~module:\\
4601       Feature~#2~of~type~#1\\
4602       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4603     }
4604     \msg_error:nn{stex}{error/nomodule}
4605   }
4606
4607   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4608
4609   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4610
4611   \stex_if_do_html:T {
4612     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4613     \stex_annotate_invisible:nnn{header}{\{ #3 }
4614   }
4615   }{
4616     \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4617     \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4618     \stex_debug:nn{features}{
4619       Feature: \l_stex_last_feature_str
4620     }
4621     \stex_if_do_html:T {
4622       \end{stex_annotate_env}
4623     }
4624   }

```

31.3 Structure

structure

```

4625 <@@=stex_structures>
4626 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4627   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4628     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4629   }
4630   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4631   {#1}{#2}
4632 }
4633

```



```

4634 \keys_define:nn { stex / features / structure } {
4635   name          .str_set_x:N = \l__stex_structures_name_str ,
4636 }
4637
4638 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4639   \str_clear:N \l__stex_structures_name_str
4640   \keys_set:nn { stex / features / structure } { #1 }
4641 }
4642
4643 \NewDocumentEnvironment{mathstructure}{m O{}}{
4644   \__stex_structures_structure_args:n { #2 }
4645   \str_if_empty:NT \l__stex_structures_name_str {
4646     \str_set:Nx \l__stex_structures_name_str { #1 }
4647   }
4648   \stex_suppress_html:n {
4649     \bool_set_true:N \l_stex_symdecl_make_macro_bool
4650     \exp_args:Nx \stex_symdecl_do:nn {
4651       name = \l__stex_structures_name_str ,
4652       def = {\STEXsymbol{module-type}}{
4653         \STEXInternalTermMathOMSiiii {
4654           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4655             { ns } ?
4656           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4657             { name } / \l__stex_structures_name_str - structure
4658         }{}{0}{}
4659       }}
4660     }{ #1 }
4661   }
4662   \exp_args:Nnnx
4663   \begin{structural_feature_module}{ structure }
4664     { \l__stex_structures_name_str }{}
4665   \stex_smsmode_do:
4666 }{
4667   \end{structural_feature_module}
4668   \stex_reset_up_to_module:n \l_stex_last_feature_str
4669   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4670   \seq_clear:N \l_tmpa_seq
4671   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4672     \seq_map_inline:cn{c_stex_module_##1_constants}{
4673       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4674     }
4675   }
4676   \exp_args:Nnno
4677   \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4678   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4679   \stex_add_structure_to_current_module:nn
4680     \l__stex_structures_name_str
4681     \l_stex_last_feature_str
4682
4683   \stex_execute_in_module:x {
4684     \tl_set:cn { #1 }{
4685       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4686     }
4687   }

```

```

4688 }
4689
4690 \cs_new:Nn \stex_invoke_structure:nn {
4691   \stex_invoke_symbol:n { #1?#2 }
4692 }
4693
4694 \cs_new_protected:Nn \stex_get_structure:n {
4695   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4696     \tl_set:Nn \l_tmpa_tl { #1 }
4697     \__stex_structures_get_from_cs:
4698   }{
4699     \cs_if_exist:cTF { #1 }{
4700       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4701       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4702       \str_if_empty:NTF \l_tmpa_str {
4703         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4704           \__stex_structures_get_from_cs:
4705         }{
4706           \__stex_structures_get_from_string:n { #1 }
4707         }
4708       }{
4709         \__stex_structures_get_from_string:n { #1 }
4710       }
4711     }{
4712       \__stex_structures_get_from_string:n { #1 }
4713     }
4714   }
4715 }
4716
4717 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4718   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4719     { \tl_tail:N \l_tmpa_tl }
4720   \str_set:Nx \l_tmpa_str {
4721     \exp_after:wN \use_i:nn \l_tmpa_tl
4722   }
4723   \str_set:Nx \l_tmpb_str {
4724     \exp_after:wN \use_ii:nn \l_tmpa_tl
4725   }
4726   \str_set:Nx \l_stex_get_structure_str {
4727     \l_tmpa_str ? \l_tmpb_str
4728   }
4729   \str_set:Nx \l_stex_get_structure_module_str {
4730     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4731   }
4732 }
4733
4734 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4735   \tl_set:Nn \l_tmpa_tl {
4736     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4737   }
4738   \str_set:Nn \l_tmpa_str { #1 }
4739   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4740
4741   \seq_map_inline:Nn \l_stex_all_modules_seq {

```

```

4742 \prop_if_exist:cT {c_stex_module_##1_structures} {
4743 \prop_map_inline:cn {c_stex_module_##1_structures} {
4744 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4745 \prop_map_break:n{\seq_map_break:n{
4746 \tl_set:Nn \l_tmpa_tl {
4747 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4748 \str_set:Nn \l_stex_get_structure_module_str {####2}
4749 }
4750 }}
4751 }
4752 }
4753 }
4754 }
4755 \l_tmpa_tl
4756 }

```

\instantiate

```

4757
4758 \keys_define:nn { stex / instantiate } {
4759 name .str_set_x:N = \l__stex_structures_name_str
4760 }
4761 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4762 \str_clear:N \l__stex_structures_name_str
4763 \keys_set:nn { stex / instantiate } { #1 }
4764 }
4765
4766 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4767 \beginingroup
4768 \stex_get_structure:n {#3}
4769 \__stex_structures_instantiate_args:n { #2 }
4770 \str_if_empty:NT \l__stex_structures_name_str {
4771 \str_set:Nn \l__stex_structures_name_str { #1 }
4772 }
4773 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4774 \seq_clear:N \l__stex_structures_fields_seq
4775 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4776 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4777 \seq_map_inline:cn {c_stex_module_##1_constants}{
4778 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4779 }
4780 }
4781
4782 \tl_if_empty:nF{#5}{
4783 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4784 \prop_clear:N \l_tmpa_prop
4785 \seq_map_inline:Nn \l_tmpa_seq {
4786 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4787 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4788 \msg_error:nnn{stex}{error/keyval}{##1}
4789 }
4790 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4791 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4792 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4793 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

```

```

4794     \exp_args:Nxx \str_if_eq:nnF
4795     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4796     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4797     \msg_error:nnxxxx{stex}{error/incompatible}
4798     {l__stex_structures_dom_str}
4799     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4800     {\l_stex_get_symbol_uri_str}
4801     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4802   }
4803   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4804 }
4805 }
4806
4807 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4808   \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4809   \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4810
4811   \stex_add_constant_to_current_module:n {\l_tmpa_str}
4812   \stex_execute_in_module:x {
4813     \prop_set_from_keyval:cn { l_stex_symdecl \l_stex_current_module_str?\l_tmpa_str _p
4814     name   = \l_tmpa_str ,
4815     args   = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4816     arity  = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4817     assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4818   }
4819   \seq_clear:c {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notations}
4820 }
4821
4822 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4823   \stex_find_notation:nn{##1}{}
4824   \stex_execute_in_module:x {
4825     \seq_put_right:cn {l_stex_symdecl\l_stex_current_module_str?\l_tmpa_str _notation
4826   }
4827
4828   \stex_copy_control_sequence_ii:ccN
4829   {stex_notation\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4830   {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4831   \l_tmpa_tl
4832   \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4833
4834
4835   \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4836     \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4837     \stex_execute_in_module:x {
4838       \tl_set:cn
4839       {stex_op_notation\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4840       { \exp_args:No \exp_not:n \l_tmpa_cs}
4841     }
4842   }
4843
4844 }
4845
4846 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4847 }

```

```

4848
4849 \stex_execute_in_module:x {
4850   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4851     domain = \l_stex_get_structure_module_str ,
4852     \prop_to_keyval:N \l_tmpa_prop
4853   }
4854   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4855 }
4856 \stex_debug:nn{instantiate}{
4857   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4858   \prop_to_keyval:N \l_tmpa_prop
4859 }
4860 \exp_args:Nxx \stex_symdecl_do:nn {
4861   type={\STEXsymbol{module-type}}{
4862     \STEXInternalTermMathOMSiiii {
4863       \l_stex_get_structure_module_str
4864     }{}{0}{}
4865   }}
4866 }{\l__stex_structures_name_str}
4867 % {
4868   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4869   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4870   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4871 % }
4872 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4873 \endgroup
4874 \stex_smsmode_do:\ignorespacesandpars
4875 }
4876
4877 \cs_new_protected:Nn \stex_symbol_or_var:n {
4878   \cs_if_exist:cTF{#1}{
4879     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4880     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4881     \str_if_empty:NTF \l_tmpa_str {
4882       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4883       \stex_invoke_variable:n {
4884         \bool_set_true:N \l_stex_symbol_or_var_bool
4885         \bool_set_false:N \l_stex_instance_or_symbol_bool
4886         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4887         \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4888         \str_set:Nx \l_stex_get_symbol_uri_str {
4889           \exp_after:wN \use:n \l_tmpa_tl
4890         }
4891       }{ % TODO \stex_invoke_varinstance:n
4892         \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl } \stex_invoke_varinstance:n {
4893           \bool_set_true:N \l_stex_symbol_or_var_bool
4894           \bool_set_true:N \l_stex_instance_or_symbol_bool
4895           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4896           \tl_set:Nx \l_tmpa_tl {\exp_after:wN \use:n \l_tmpa_tl}
4897           \str_set:Nx \l_stex_get_symbol_uri_str {
4898             \exp_after:wN \use:n \l_tmpa_tl
4899           }
4900         }{
4901           \bool_set_false:N \l_stex_symbol_or_var_bool

```

```

4902         \stex_get_symbol:n{#1}
4903     }
4904 }
4905 }{
4906     \__stex_structures_symbolorvar_from_string:n{ #1 }
4907 }
4908 }{
4909     \__stex_structures_symbolorvar_from_string:n{ #1 }
4910 }
4911 }
4912
4913 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4914     \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4915         \bool_set_true:N \l_stex_symbol_or_var_bool
4916         \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4917     }{
4918         \bool_set_false:N \l_stex_symbol_or_var_bool
4919         \stex_get_symbol:n{#1}
4920     }
4921 }
4922
4923 \keys_define:nn { stex / varinstantiate } {
4924     name          .str_set_x:N = \l__stex_structures_name_str,
4925     bind          .choices:nn =
4926         {forall,exists}
4927         {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4928 }
4929 }
4930 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4931     \str_clear:N \l__stex_structures_name_str
4932     \str_clear:N \l__stex_structures_bind_str
4933     \keys_set:nn { stex / varinstantiate } { #1 }
4934 }
4935
4936 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4937     \begingroup
4938         \stex_get_structure:n {#3}
4939         \__stex_structures_varinstantiate_args:n { #2 }
4940         \str_if_empty:NT \l__stex_structures_name_str {
4941             \str_set:Nn \l__stex_structures_name_str { #1 }
4942         }
4943         \stex_if_do_html:TF{
4944             \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4945         }{\use:n}
4946         {
4947             \stex_if_do_html:T{
4948                 \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}
4949             }
4950             \seq_clear:N \l__stex_structures_fields_seq
4951             \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4952             \seq_map_inline:Nn \l_stex_collect_imports_seq {
4953                 \seq_map_inline:cn {c_stex_module_##1_constants}{
4954                     \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4955                 }

```

```

4956 }
4957 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4958 \prop_clear:N \l_tmpa_prop
4959 \tl_if_empty:nF {#5} {
4960   \seq_set_split:Nnn \l_tmpa_seq , {#5}
4961   \seq_map_inline:Nn \l_tmpa_seq {
4962     \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4963     \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4964       \msg_error:nnn{stex}{error/keyval}{##1}
4965     }
4966     \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4967     \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4968     \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_in_seq:nn
4969     \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4970     \stex_if_do_html:T{
4971       \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,
4972         \bool_if:NTF\l_stex_symbol_or_var_bool{var://}{}\l_stex_get_symbol_uri_str}{}}
4973     }
4974     \bool_if:NTF \l_stex_symbol_or_var_bool {
4975       \exp_args:Nxx \str_if_eq:nnF
4976         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4977         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{\
4978         \msg_error:nnxxx{stex}{error/incompatible}
4979         {\l__stex_structures_dom_str}
4980         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4981         {\l_stex_get_symbol_uri_str}
4982         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}}
4983     }
4984     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4985   }}{
4986     \exp_args:Nxx \str_if_eq:nnF
4987       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4988       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{\
4989       \msg_error:nnxxx{stex}{error/incompatible}
4990       {\l__stex_structures_dom_str}
4991       {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4992       {\l_stex_get_symbol_uri_str}
4993       {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}
4994     }
4995     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4996   }}
4997   }
4998 }
4999 \tl_gclear:N \g__stex_structures_aftergroup_tl
5000 \seq_map_inline:Nn \l__stex_structures_fields_seq {
5001   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1} \l_tmpa_str}
5002   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
5003   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
5004     \stex_find_notation:nn{##1}{}
5005     \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str _cs}
5006       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5007     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa\l_tmpa_str _cs}}
5008     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
5009       \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str _cs}

```

```

5010         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
5011         \stex_debug:nn{varinstantiate}{Operator-Notation:~\cs_meaning:c{g__stex_struct
5012     }
5013 }
5014
5015 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5016     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
5017         name    = \l_tmpa_str ,
5018         args    = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
5019         arity   = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
5020         assocs  = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
5021     }
5022     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
5023     {g__stex_structures_tmpa_\l_tmpa_str _cs}
5024     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
5025     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
5026 }
5027 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
5028 }
5029 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
5030     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str _prop }{
5031         domain = \l_stex_get_structure_module_str ,
5032         \prop_to_keyval:N \l_tmpa_prop
5033     }
5034     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
5035     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str _op_tl}{
5036         \exp_args:Nnx \exp_not:N \use:nn {
5037             \str_set:Nn \exp_not:N \STEXInternalCurrentSymbolStr {var://\l__stex_structures_
5038             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
5039                 \exp_not:n{
5040                     \_varcomp{#4}
5041                 }
5042             }
5043         }{
5044             \exp_not:n{\_stex_reset:N \STEXInternalCurrentSymbolStr}
5045         }
5046     }
5047 }
5048 }
5049 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
5050 \aftergroup\g__stex_structures_aftergroup_tl
5051 \endgroup
5052 \stex_smsmode_do:\ignorespacesandpars
5053 }
5054
5055 \cs_new_protected:Nn \stex_invoke_instance:n {
5056     \peek_charcode_remove:NTF ! {
5057         \stex_invoke_symbol:n{#1}
5058     }{
5059         \_stex_invoke_instance:nn {#1}
5060     }
5061 }
5062
5063

```



```

5064 \cs_new_protected:Nn \stex_invoke_varinstance:n {
5065   \peek_charcode_remove:NTF ! {
5066     \exp_args:Nnx \use:nn {
5067       \def\comp{\_varcomp}
5068       \use:c{l_stex_varinstance_#1_op_tl}
5069     }{
5070       \_stex_reset:N \comp
5071     }
5072   }{
5073     \_stex_invoke_varinstance:nn {#1}
5074   }
5075 }
5076
5077 \cs_new_protected:Nn \_stex_invoke_instance:nn {
5078   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
5079     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
5080   }{
5081     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
5082     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
5083       \prop_to_keyval:N \l_tmpa_prop
5084     }
5085   }
5086 }
5087
5088 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
5089   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
5090     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
5091     \l_tmpa_tl
5092   }{
5093     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
5094   }
5095 }

```

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

\stex_invoke_structure:nnn

```

5096 % #1: URI of the instance
5097 % #2: URI of the instantiated module
5098 \cs_new_protected:Nn \stex_invoke_structure:nnn {
5099   \tl_if_empty:nTF{ #3 }{
5100     \prop_set_eq:Nc \l__stex_structures_structure_prop {
5101       c_stex_feature_ #2 _prop
5102     }
5103     \tl_clear:N \l_tmpa_tl
5104     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
5105     \seq_map_inline:Nn \l_tmpa_seq {
5106       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
5107       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
5108       \cs_if_exist:cT {
5109         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
5110       }{
5111         \tl_if_empty:NF \l_tmpa_tl {
5112           \tl_put_right:Nn \l_tmpa_tl {,}
5113         }

```

```

5114         \tl_put_right:Nx \l_tmpa_tl {
5115             \stex_invoke_symbol:n {#1/\l_tmpa_str}!
5116         }
5117     }
5118 }
5119 \exp_args:No \mathstruct \l_tmpa_tl
5120 }{
5121     \stex_invoke_symbol:n{#1/#3}
5122 }
5123 }

```

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

```

5124 </package>

```

Chapter 32

STEX -Statements Implementation

```
5125 <*package>
5126
5127 %%%%%%%%%%% features.dtx %%%%%%%%%%%
5128
5129 <@@=stex_statements>

Warnings and error messages

5130

\titleemph

5131 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
5132 \keys_define:nn {stex / definiendum }{
5133   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
5134   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
5135   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
5136   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
5137 }
5138 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
5139   \str_clear:N \l__stex_statements_definiendum_root_str
5140   \tl_clear:N \l__stex_statements_definiendum_post_tl
5141   \str_clear:N \l__stex_statements_definiendum_gfa_str
5142   \keys_set:nn { stex / definiendum }{ #1 }
5143 }
5144 \NewDocumentCommand \definiendum { O{} m m } {
5145   \__stex_statements_definiendum_args:n { #1 }
5146   \stex_get_symbol:n { #2 }
5147   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5148   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
5149     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

5150     \tl_set:Nn \l_tmpa_tl { #3 }
5151   } {
5152     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
5153     \tl_set:Nn \l_tmpa_tl {
5154       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
5155     }
5156   }
5157 } {
5158   \tl_set:Nn \l_tmpa_tl { #3 }
5159 }
5160
5161 % TODO root
5162 \stex_html_backend:TF {
5163   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
5164 } {
5165   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
5166 }
5167 }
5168 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

5169
5170 \NewDocumentCommand \definame { 0{ } m } {
5171   \__stex_statements_definiendum_args:n { #1 }
5172   % TODO: root
5173   \stex_get_symbol:n { #2 }
5174   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5175   \str_set:Nx \l_tmpa_str {
5176     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5177   }
5178   \str_replace_all:Nnn \l_tmpa_str {-} {~}
5179   \stex_html_backend:TF {
5180     \stex_if_do_html:T {
5181       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5182         \l_tmpa_str\l__stex_statements_definiendum_post_tl
5183       }
5184     }
5185   } {
5186     \exp_args:Nnx \defemph@uri {
5187       \l_tmpa_str\l__stex_statements_definiendum_post_tl
5188     } { \l_stex_get_symbol_uri_str }
5189   }
5190 }
5191 \stex_deactivate_macro:Nn \definame {definition~environments}
5192
5193 \NewDocumentCommand \Definame { 0{ } m } {
5194   \__stex_statements_definiendum_args:n { #1 }
5195   \stex_get_symbol:n { #2 }
5196   \str_set:Nx \l_tmpa_str {
5197     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
5198   }
5199   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

5200 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5201 \stex_html_backend:TF {
5202   \stex_if_do_html:T {
5203     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
5204       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5205     }
5206   }
5207 } {
5208   \exp_args:Nnx \defemph@uri {
5209     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
5210   } { \l_stex_get_symbol_uri_str }
5211 }
5212 }
5213 \stex_deactivate_macro:Nn \Definame {definition-environments}
5214
5215 \NewDocumentCommand \premise { m }{
5216   \noindent\stex_annotate:nnn{ premise }{}{\ignorespaces #1 }
5217 }
5218 \NewDocumentCommand \conclusion { m }{
5219   \noindent\stex_annotate:nnn{ conclusion }{}{\ignorespaces #1 }
5220 }
5221 \NewDocumentCommand \definiens { 0{} m }{
5222   \str_clear:N \l_stex_get_symbol_uri_str
5223   \tl_if_empty:nF {#1} {
5224     \stex_get_symbol:n { #1 }
5225   }
5226   \str_if_empty:NT \l_stex_get_symbol_uri_str {
5227     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5228       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5229     }{
5230       % TODO throw error
5231     }
5232   }
5233   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5234   {\l_stex_current_module_str}{
5235     \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5236   }{true}{
5237     \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5238     \exp_args:Nx \stex_add_to_current_module:n {
5239       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5240     }
5241   }
5242 }
5243 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5244 }
5245
5246 \NewDocumentCommand \varbindforall {m}{
5247   \stex_symbol_or_var:n {#1}
5248   \bool_if:NTF\l_stex_symbol_or_var_bool{
5249     \stex_if_do_html:T {
5250       \stex_annotate_invisible:nnn {bindtype}{forall,\l_stex_get_symbol_uri_str}{}
5251     }
5252   }{
5253     % todo throw error

```

```

5254 }
5255 }
5256
5257 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5258 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5259 \stex_deactivate_macro:Nn \definiens {definition~environments}
5260 \stex_deactivate_macro:Nn \varbindforall {definition~or~assertion~environments}
5261

```

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

sdefinition

```

5262
5263 \keys_define:nn {stex / sdefinition }{
5264   type      .str_set_x:N = \sdefinitiontype,
5265   id        .str_set_x:N = \sdefinitionid,
5266   name      .str_set_x:N = \sdefinitionname,
5267   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5268   title     .tl_set:N     = \sdefinitiontitle
5269 }
5270 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5271   \str_clear:N \sdefinitiontype
5272   \str_clear:N \sdefinitionid
5273   \str_clear:N \sdefinitionname
5274   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5275   \tl_clear:N \sdefinitiontitle
5276   \keys_set:nn { stex / sdefinition }{ #1 }
5277 }
5278
5279 \NewDocumentEnvironment{sdefinition}{0{}}{
5280   \__stex_statements_sdefinition_args:n{ #1 }
5281   \stex_reactivate_macro:N \definiendum
5282   \stex_reactivate_macro:N \definame
5283   \stex_reactivate_macro:N \Definame
5284   \stex_reactivate_macro:N \premise
5285   \stex_reactivate_macro:N \definiens
5286   \stex_reactivate_macro:N \varbindforall
5287   \stex_if_smsmode:F{
5288     \seq_clear:N \l_tmpb_seq
5289     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5290       \tl_if_empty:nF{ ##1 }{
5291         \stex_get_symbol:n { ##1 }
5292         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5293           \l_stex_get_symbol_uri_str
5294         }
5295       }
5296     }
5297     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5298     \exp_args:Nnnx
5299     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
5300     \str_if_empty:NF \sdefinitiontype {
5301       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{ }
5302     }
5303     \str_if_empty:NF \sdefinitionname {

```

```

5304     \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5305   }
5306   \clist_set:No \l_tmpa_clist \sdefinitiontype
5307   \tl_clear:N \l_tmpa_tl
5308   \clist_map_inline:Nn \l_tmpa_clist {
5309     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5310       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5311     }
5312   }
5313   \tl_if_empty:NTF \l_tmpa_tl {
5314     \__stex_statements_sdefinition_start:
5315   }{
5316     \l_tmpa_tl
5317   }
5318 }
5319 \stex_ref_new_doc_target:n \sdefinitionid
5320 \stex_smsmode_do:
5321 }{
5322   \stex_suppress_html:n {
5323     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5324   }
5325   \stex_if_smsmode:F {
5326     \clist_set:No \l_tmpa_clist \sdefinitiontype
5327     \tl_clear:N \l_tmpa_tl
5328     \clist_map_inline:Nn \l_tmpa_clist {
5329       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5330         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5331       }
5332     }
5333     \tl_if_empty:NTF \l_tmpa_tl {
5334       \__stex_statements_sdefinition_end:
5335     }{
5336       \l_tmpa_tl
5337     }
5338     \end{stex_annotate_env}
5339   }
5340 }

```

\stexpatchdefinition

```

5341 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5342   \stex_par:\noindent\titleemph{Definition}\tl_if_empty:NF \sdefinitiontitle {
5343     ~(\sdefinitiontitle)
5344   }~}
5345 }
5346 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\stex_par:\medskip}
5347
5348 \newcommand\stexpatchdefinition[3]{} {
5349   \str_set:Nx \l_tmpa_str{ #1 }
5350   \str_if_empty:NTF \l_tmpa_str {
5351     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5352     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5353   }{
5354     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5355     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }

```

```

5356     }
5357 }

```

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

`\inlinedef` inline:

```

5358 \keys_define:nn {stex / inlinedef }{
5359   type      .str_set_x:N = \sdefinitiontype,
5360   id        .str_set_x:N = \sdefinitionid,
5361   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5362   name      .str_set_x:N = \sdefinitionname
5363 }
5364 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5365   \str_clear:N \sdefinitiontype
5366   \str_clear:N \sdefinitionid
5367   \str_clear:N \sdefinitionname
5368   \clist_clear:N \l__stex_statements_sdefinition_for_clist
5369   \keys_set:nn { stex / inlinedef }{ #1 }
5370 }
5371 \NewDocumentCommand \inlinedef { 0{} m } {
5372   \begingroup
5373   \__stex_statements_inlinedef_args:n{ #1 }
5374   \stex_reactivate_macro:N \definiendum
5375   \stex_reactivate_macro:N \definame
5376   \stex_reactivate_macro:N \Definame
5377   \stex_reactivate_macro:N \premise
5378   \stex_reactivate_macro:N \definiens
5379   \stex_reactivate_macro:N \varbindforall
5380   \stex_ref_new_doc_target:n \sdefinitionid
5381   \stex_if_smsmode:TF{\stex_suppress_html:n {
5382     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5383   }}{
5384     \seq_clear:N \l_tmpb_seq
5385     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5386       \tl_if_empty:nF{ ##1 }{
5387         \stex_get_symbol:n { ##1 }
5388         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5389           \l_stex_get_symbol_uri_str
5390         }
5391       }
5392     }
5393     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpb_seq
5394     \exp_args:Nnx
5395     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpb_seq {,}}{
5396       \str_if_empty:NF \sdefinitiontype {
5397         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5398       }
5399       #2
5400       \str_if_empty:NF \sdefinitionname {
5401         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5402         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5403       }
5404     }
5405   }

```



```

5453 \clist_map_inline:Nn \l_tmpa_clist {
5454   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5455     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5456   }
5457 }
5458 \tl_if_empty:NTF \l_tmpa_tl {
5459   \__stex_statements_sassertion_start:
5460 }{
5461   \l_tmpa_tl
5462 }
5463 }
5464 \str_if_empty:NTF \sassertionid {
5465   \str_if_empty:NF \sassertionname {
5466     \stex_ref_new_doc_target:n {}
5467   }
5468 } {
5469   \stex_ref_new_doc_target:n \sassertionid
5470 }
5471 \stex_smsmode_do:
5472 ){
5473   \str_if_empty:NF \sassertionname {
5474     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5475     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5476   }
5477   \stex_if_smsmode:F {
5478     \clist_set:No \l_tmpa_clist \sassertiontype
5479     \tl_clear:N \l_tmpa_tl
5480     \clist_map_inline:Nn \l_tmpa_clist {
5481       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5482         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5483       }
5484     }
5485     \tl_if_empty:NTF \l_tmpa_tl {
5486       \__stex_statements_sassertion_end:
5487     }{
5488       \l_tmpa_tl
5489     }
5490     \end{stex_annotate_env}
5491   }
5492 }

```

\stexpatchassertion

```

5493
5494 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5495   \stex_par:\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5496     (\sassertiontitle)
5497   }~}
5498 }
5499 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\stex_par:\medskip}
5500
5501 \newcommand\stexpatchassertion[3] [] {
5502   \str_set:Nx \l_tmpa_str{ #1 }
5503   \str_if_empty:NTF \l_tmpa_str {
5504     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }

```

```

5505     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5506   }{
5507     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5508     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5509   }
5510 }

```

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

`\inlineass` inline:

```

5511 \keys_define:nn {stex / inlineass }{
5512   type      .str_set_x:N = \sassertiontype,
5513   id        .str_set_x:N = \sassertionid,
5514   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5515   name      .str_set_x:N = \sassertionname
5516 }
5517 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5518   \str_clear:N \sassertiontype
5519   \str_clear:N \sassertionid
5520   \str_clear:N \sassertionname
5521   \clist_clear:N \l__stex_statements_sassertion_for_clist
5522   \keys_set:nn { stex / inlineass }{ #1 }
5523 }
5524 \NewDocumentCommand \inlineass { 0{} m } {
5525   \beginngroup
5526   \stex_reactivate_macro:N \premise
5527   \stex_reactivate_macro:N \conclusion
5528   \stex_reactivate_macro:N \varbindforall
5529   \__stex_statements_inlineass_args:n{ #1 }
5530   \str_if_empty:NTF \sassertionid {
5531     \str_if_empty:NF \sassertionname {
5532       \stex_ref_new_doc_target:n {}
5533     }
5534   } {
5535     \stex_ref_new_doc_target:n \sassertionid
5536   }
5537
5538   \stex_if_smsmode:TF{
5539     \str_if_empty:NF \sassertionname {
5540       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5541     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5542   }
5543 }{
5544   \seq_clear:N \l_tmpb_seq
5545   \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5546     \tl_if_empty:nF{ ##1 }{
5547       \stex_get_symbol:n { ##1 }
5548       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5549         \l_stex_get_symbol_uri_str
5550       }
5551     }
5552   }
5553   \exp_args:Nnx
5554   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpb_seq {},,}{

```

```

5555     \str_if_empty:NF \sassassertiontype {
5556       \stex_annotate_invisible:nnn{typestrings}{\sassassertiontype}{}
5557     }
5558     #2
5559     \str_if_empty:NF \sassassertionname {
5560       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassassertionname}}
5561       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassassertionname}
5562       \stex_annotate_invisible:nnn{statementname}{\sassassertionname}{}
5563     }
5564   }
5565 }
5566 \endgroup
5567 \stex_smsmode_do:
5568 }

```

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

32.3 Examples

`sexample`

```

5569
5570 \keys_define:nn {stex / sexample }{
5571   type      .str_set_x:N = \exampletype,
5572   id        .str_set_x:N = \sexampleid,
5573   title     .tl_set:N    = \sexamplename,
5574   name      .str_set_x:N = \sexamplename ,
5575   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5576 }
5577 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5578   \str_clear:N \sexampletype
5579   \str_clear:N \sexampleid
5580   \str_clear:N \sexamplename
5581   \tl_clear:N \sexamplename
5582   \clist_clear:N \l__stex_statements_sexample_for_clist
5583   \keys_set:nn { stex / sexample }{ #1 }
5584 }
5585
5586 \NewDocumentEnvironment{sexample}{0{}}{
5587   \__stex_statements_sexample_args:n{ #1 }
5588   \stex_reactivate_macro:N \premise
5589   \stex_reactivate_macro:N \conclusion
5590   \stex_if_smsmode:F {
5591     \seq_clear:N \l_tmpb_seq
5592     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5593       \tl_if_empty:nF{ ##1 }{
5594         \stex_get_symbol:n { ##1 }
5595         \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5596           \l_stex_get_symbol_uri_str
5597         }
5598       }
5599     }
5600     \exp_args:Nnnx
5601     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}

```

```

5602 \str_if_empty:NF \sexamplotype {
5603   \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5604 }
5605 \str_if_empty:NF \sexamplename {
5606   \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5607 }
5608 \clist_set:No \l_tmpa_clist \sexamplotype
5609 \tl_clear:N \l_tmpa_tl
5610 \clist_map_inline:Nn \l_tmpa_clist {
5611   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5612     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5613   }
5614 }
5615 \tl_if_empty:NTF \l_tmpa_tl {
5616   \__stex_statements_sexample_start:
5617 }{
5618   \l_tmpa_tl
5619 }
5620 }
5621 \str_if_empty:NF \sexampleid {
5622   \stex_ref_new_doc_target:n \sexampleid
5623 }
5624 \stex_smsmode_do:
5625 }{
5626   \str_if_empty:NF \sexamplename {
5627     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5628   }
5629   \stex_if_smsmode:F {
5630     \clist_set:No \l_tmpa_clist \sexamplotype
5631     \tl_clear:N \l_tmpa_tl
5632     \clist_map_inline:Nn \l_tmpa_clist {
5633       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5634         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5635       }
5636     }
5637     \tl_if_empty:NTF \l_tmpa_tl {
5638       \__stex_statements_sexample_end:
5639     }{
5640       \l_tmpa_tl
5641     }
5642     \end{stex_annotate_env}
5643   }
5644 }

```

\stexpatchexample

```

5645
5646 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5647   \stex_par:\noindent\titllemph{Example~\tl_if_empty:NF \sexampltitle {
5648     (\sexampltitle)
5649   }~}
5650 }
5651 \cs_new_protected:Nn \__stex_statements_sexample_end: {\stex_par:\medskip}
5652
5653 \newcommand\stexpatchexample[3]{} {

```

```

5654 \str_set:Nx \l_tmpa_str{ #1 }
5655 \str_if_empty:NTF \l_tmpa_str {
5656   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5657   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5658 }{
5659   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5660   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5661 }
5662 }

```

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

`\inlineex` inline:

```

5663 \keys_define:nn {stex / inlineex }{
5664   type      .str_set_x:N = \sexamplotype,
5665   id        .str_set_x:N = \sexampleid,
5666   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5667   name      .str_set_x:N = \sexamplename
5668 }
5669 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5670   \str_clear:N \sexamplotype
5671   \str_clear:N \sexampleid
5672   \str_clear:N \sexamplename
5673   \clist_clear:N \l__stex_statements_sexample_for_clist
5674   \keys_set:nn { stex / inlineex }{ #1 }
5675 }
5676 \NewDocumentCommand \inlineex { 0{} m } {
5677   \begingroup
5678   \stex_reactivate_macro:N \premise
5679   \stex_reactivate_macro:N \conclusion
5680   \__stex_statements_inlineex_args:n{ #1 }
5681   \str_if_empty:NF \sexampleid {
5682     \stex_ref_new_doc_target:n \sexampleid
5683   }
5684   \stex_if_smsmode:TF{
5685     \str_if_empty:NF \sexamplename {
5686       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5687   }
5688 }{
5689   \seq_clear:N \l_tmpb_seq
5690   \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5691     \tl_if_empty:nF{ ##1 }{
5692       \stex_get_symbol:n { ##1 }
5693       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5694         \l_stex_get_symbol_uri_str
5695       }
5696     }
5697   }
5698   \exp_args:Nnx
5699   \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpb_seq {,}}{
5700     \str_if_empty:NF \sexamplotype {
5701       \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5702     }
5703     #2

```

```

5704     \str_if_empty:NF \sexamplename {
5705       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sexamplename}}
5706       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5707     }
5708   }
5709 }
5710 \endgroup
5711 \stex_smsmode_do:
5712 }

```

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

32.4 Logical Paragraphs

`sparagraph`

```

5713 \keys_define:nn { stex / sparagraph } {
5714   id      .str_set:x:N = \sparagraphid ,
5715   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5716   type    .str_set:x:N = \sparagraphtype ,
5717   for     .clist_set:N  = \l__stex_statements_sparagraph_for_clist ,
5718   from    .tl_set:N    = \sparagraphfrom ,
5719   to      .tl_set:N    = \sparagraphto ,
5720   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5721   name    .str_set:N    = \sparagraphname ,
5722   imports .tl_set:N    = \l__stex_statements_sparagraph_imports_tl
5723 }
5724
5725 \cs_new_protected:Nn \stex_sparagraph_args:n {
5726   \tl_clear:N \l_stex_sparagraph_title_tl
5727   \tl_clear:N \sparagraphfrom
5728   \tl_clear:N \sparagraphto
5729   \tl_clear:N \l_stex_sparagraph_start_tl
5730   \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5731   \str_clear:N \sparagraphid
5732   \str_clear:N \sparagraphtype
5733   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5734   \str_clear:N \sparagraphname
5735   \keys_set:nn { stex / sparagraph } { #1 }
5736 }
5737 \newif\if@in@omtext\@in@omtextfalse
5738
5739 \NewDocumentEnvironment {sparagraph} { 0{ } } {
5740   \stex_sparagraph_args:n { #1 }
5741   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5742     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5743   }{
5744     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5745   }
5746   \@in@omtexttrue
5747   \stex_if_smsmode:F {
5748     \seq_clear:N \l_tmpb_seq
5749     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5750       \tl_if_empty:nF{ ##1 }{

```

```

5751     \stex_get_symbol:n { ##1 }
5752     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5753         \l_stex_get_symbol_uri_str
5754     }
5755 }
5756 }
5757 \exp_args:Nnnx
5758 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}
5759 \str_if_empty:NF \sparagraphtype {
5760     \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5761 }
5762 \str_if_empty:NF \sparaagraphfrom {
5763     \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5764 }
5765 \str_if_empty:NF \sparaagraphto {
5766     \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5767 }
5768 \str_if_empty:NF \sparaagraphname {
5769     \stex_annotate_invisible:nnn{statementname}{\sparaagraphname}{}
5770 }
5771 \clist_set:No \l_tmpa_clist \sparaagraphtype
5772 \tl_clear:N \l_tmpa_tl
5773 \clist_map_inline:Nn \sparaagraphtype {
5774     \tl_if_exist:cT {__stex_statements_sparaagraph_##1_start:}{
5775         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparaagraph_##1_start:}}
5776     }
5777 }
5778 \stex_csl_to_imports:No \usemodule \l__stex_statements_sparaagraph_imports_tl
5779 \tl_if_empty:NTF \l_tmpa_tl {
5780     \__stex_statements_sparaagraph_start:
5781 }{
5782     \l_tmpa_tl
5783 }
5784 }
5785 \clist_set:No \l_tmpa_clist \sparaagraphtype
5786 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5787     {
5788         \stex_reactivate_macro:N \definiendum
5789         \stex_reactivate_macro:N \definame
5790         \stex_reactivate_macro:N \Definame
5791         \stex_reactivate_macro:N \premise
5792         \stex_reactivate_macro:N \definiens
5793     }
5794 \str_if_empty:NTF \sparaagraphid {
5795     \str_if_empty:NTF \sparaagraphname {
5796         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5797             \stex_ref_new_doc_target:n {}
5798         }
5799     } {
5800         \stex_ref_new_doc_target:n {}
5801     }
5802 } {
5803     \stex_ref_new_doc_target:n \sparaagraphid
5804 }

```



```

5805 \exp_args:NNx
5806 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5807   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5808     \tl_if_empty:nF{ ##1 }{
5809       \stex_get_symbol:n { ##1 }
5810       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5811     }
5812   }
5813 }
5814 \stex_smsmode_do:
5815 \ignorespacesandpars
5816 }{
5817   \str_if_empty:NF \sparagraphname {
5818     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5819     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5820   }
5821   \stex_if_smsmode:F {
5822     \clist_set:No \l_tmpa_clist \sparagraphtype
5823     \tl_clear:N \l_tmpa_tl
5824     \clist_map_inline:Nn \l_tmpa_clist {
5825       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5826         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5827       }
5828     }
5829     \tl_if_empty:NTF \l_tmpa_tl {
5830       \__stex_statements_sparagraph_end:
5831     }{
5832       \l_tmpa_tl
5833     }
5834     \end{stex_annotate_env}
5835   }
5836 }

```

\stexpatchparagraph

```

5837
5838 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5839   \stex_par:\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5840     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5841       \titleemph{\l_stex_sparagraph_title_tl}:~
5842     }
5843   }{
5844     \titleemph{\l_stex_sparagraph_start_tl}~
5845   }
5846 }
5847 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\stex_par:\medskip}
5848
5849 \newcommand\stexpatchparagraph[3] [] {
5850   \str_set:Nx \l_tmpa_str{ #1 }
5851   \str_if_empty:NTF \l_tmpa_str {
5852     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5853     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5854   }{
5855     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5856     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }

```

```

5857     }
5858 }
5859
5860 \keys_define:nn { stex / inlinepara } {
5861   id      .str_set:N = \sparagraphid ,
5862   type    .str_set:N = \sparagraphtype ,
5863   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5864   from    .tl_set:N   = \sparagraphfrom ,
5865   to      .tl_set:N   = \sparagraphto ,
5866   name    .str_set:N   = \sparagraphname
5867 }
5868 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5869   \tl_clear:N \sparagraphfrom
5870   \tl_clear:N \sparagraphto
5871   \str_clear:N \sparagraphid
5872   \str_clear:N \sparagraphtype
5873   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5874   \str_clear:N \sparagraphname
5875   \keys_set:nn { stex / inlinepara }{ #1 }
5876 }
5877 \NewDocumentCommand \inlinepara { 0{} m } {
5878   \begingroup
5879   \__stex_statements_inlinepara_args:n{ #1 }
5880   \clist_set:Nn \l_tmpa_clist \sparagraphtype
5881   \str_if_empty:NTF \sparagraphid {
5882     \str_if_empty:NTF \sparagraphname {
5883       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5884         \stex_ref_new_doc_target:n {}
5885       }
5886     } {
5887       \stex_ref_new_doc_target:n {}
5888     }
5889   } {
5890     \stex_ref_new_doc_target:n \sparagraphid
5891   }
5892   \stex_if_smsmode:TF{
5893     \str_if_empty:NF \sparagraphname {
5894       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5895     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5896   }
5897 }{
5898   \seq_clear:N \l_tmpb_seq
5899   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5900     \tl_if_empty:nF{ ##1 }{
5901       \stex_get_symbol:n { ##1 }
5902       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
5903         \l_stex_get_symbol_uri_str
5904       }
5905     }
5906   }
5907   \exp_args:Nnx
5908   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpb_seq {,}}{
5909     \str_if_empty:NF \sparagraphtype {
5910       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{

```

```

5911     }
5912     \str_if_empty:NF \sparagraphfrom {
5913       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5914     }
5915     \str_if_empty:NF \sparagraphto {
5916       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5917     }
5918     \str_if_empty:NF \sparagraphname {
5919       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5920       \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5921       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5922     }
5923     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5924       \clist_map_inline:Nn \l_tmpb_seq {
5925         \stex_ref_new_sym_target:n {##1}
5926       }
5927     }
5928     #2
5929   }
5930 }
5931 \endgroup
5932 \stex_smsmode_do:
5933 }
5934

```

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

```

5935 </package>

```

Chapter 33

The Implementation

```
5936 <*package>
5937 <@@=stex_sproof>
5938
5939 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5940
```

33.1 Proofs

We first define some keys for the proof environment.

```
5941 \keys_define:nn { stex / spf } {
5942   id          .str_set_x:N = \spfid,
5943   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5944   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5945   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5946   type        .str_set_x:N = \spftype,
5947   title       .tl_set:N    = \spftitle,
5948   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5949   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5950   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5951 }
5952 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5953   \str_clear:N \spfid
5954   \tl_clear:N \l__stex_sproof_spf_for_tl
5955   \tl_clear:N \l__stex_sproof_spf_from_tl
5956   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5957   \str_clear:N \spftype
5958   \tl_clear:N \spftitle
5959   \tl_clear:N \l__stex_sproof_spf_continues_tl
5960   \tl_clear:N \l__stex_sproof_spf_functions_tl
5961   \tl_clear:N \l__stex_sproof_spf_method_tl
5962   \bool_set_false:N \l__stex_sproof_inc_counter_bool
5963   \keys_set:nn { stex / spf }{ #1 }
5964 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5965 \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.

```

5966 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5967 \cs_new_protected:Npn \sproofnumber {
5968   \int_set:Nn \l_tmpa_int {1}
5969   \bool_while_do:nn {
5970     \int_compare_p:nNn {
5971       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5972     } > 0
5973   }{
5974     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5975     \int_incr:N \l_tmpa_int
5976   }
5977 }
5978 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5979   \int_set:Nn \l_tmpa_int {1}
5980   \bool_while_do:nn {
5981     \int_compare_p:nNn {
5982       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5983     } > 0
5984   }{
5985     \int_incr:N \l_tmpa_int
5986   }
5987   \int_compare:nNnF \l_tmpa_int = 1 {
5988     \int_decr:N \l_tmpa_int
5989   }
5990   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5991     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5992   }
5993 }
5994
5995 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5996   \int_set:Nn \l_tmpa_int {1}
5997   \bool_while_do:nn {
5998     \int_compare_p:nNn {
5999       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6000     } > 0
6001   }{
6002     \int_incr:N \l_tmpa_int
6003   }
6004   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
6005 }
6006
6007 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
6008   \int_set:Nn \l_tmpa_int {1}
6009   \bool_while_do:nn {

```

```

6010 \int_compare_p:nNn {
6011 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
6012 } > 0
6013 }{
6014 \int_incr:N \l_tmpa_int
6015 }
6016 \int_decr:N \l_tmpa_int
6017 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
6018 }

```

\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

```

6019 \def\sproof@box{
6020 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
6021 }
6022 \def\sproofend{
6023 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
6024 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
6025 }
6026 }

```

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

spf@*@kw

```

6027 \def\spf@proofsketch@kw{Proof~Sketch}
6028 \def\spf@proof@kw{Proof}
6029 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

6030 \AddToHook{begindocument}{
6031 \ltx@ifpackageloaded{babel}{
6032 \makeatletter
6033 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6034 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6035 \input{sproof-ngerman.ldf}
6036 }
6037 \clist_if_in:NnT \l_tmpa_clist {finnish}{
6038 \input{sproof-finnish.ldf}
6039 }
6040 \clist_if_in:NnT \l_tmpa_clist {french}{
6041 \input{sproof-french.ldf}
6042 }
6043 \clist_if_in:NnT \l_tmpa_clist {russian}{
6044 \input{sproof-russian.ldf}
6045 }
6046 \makeatother
6047 }{}
6048 }

```

spfsketch

```

6049 \newcommand\spsketch[2] [] {
6050 \begin{group}
6051 \let \premise \stex_proof_premise:

```

```

6052 \__stex_sproof_spf_args:n{#1}
6053 \stex_if_smsmode:TF {
6054   \str_if_empty:NF \spfid {
6055     \stex_ref_new_doc_target:n \spfid
6056   }
6057 }{
6058   \seq_clear:N \l_tmpa_seq
6059   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6060     \tl_if_empty:nF{ ##1 }{
6061       \stex_get_symbol:n { ##1 }
6062       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6063         \l_stex_get_symbol_uri_str
6064       }
6065     }
6066   }
6067   \exp_args:Nnx
6068   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
6069     \str_if_empty:NF \spftype {
6070       \stex_annotate_invisible:nnn{type}{\spftype}{
6071       }
6072       \clist_set:Nn \l_tmpa_clist \spftype
6073       \tl_set:Nn \l_tmpa_tl {
6074         \titleemph{
6075           \tl_if_empty:NTF \spftitle {
6076             \spf@proofsketch@kw
6077           }{
6078             \spftitle
6079           }
6080         }:\sim
6081       }
6082       \clist_map_inline:Nn \l_tmpa_clist {
6083         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6084           \tl_clear:N \l_tmpa_tl
6085         }
6086       }
6087       \str_if_empty:NF \spfid {
6088         \stex_ref_new_doc_target:n \spfid
6089       }
6090       \l_tmpa_tl #2 \sproofend
6091     }
6092   }
6093   \endgroup
6094   \stex_smsmode_do:
6095 }
6096

```

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

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

```

6097 \newenvironment{spfeq}[2][ ]{
6098   \__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

```

6099 \let \premise \stex_proof_premise:
6100 \stex_if_smsmode:TF {
6101   \str_if_empty:NF \spfid {
6102     \stex_ref_new_doc_target:n \spfid
6103   }
6104 }{
6105   \seq_clear:N \l_tmpa_seq
6106   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6107     \tl_if_empty:nF{ ##1 }{
6108       \stex_get_symbol:n { ##1 }
6109       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6110         \l_stex_get_symbol_uri_str
6111       }
6112     }
6113   }
6114   \exp_args:Nnnx
6115   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
6116   \str_if_empty:NF \spftype {
6117     \stex_annotate_invisible:nnn{type}{\spftype}{}
6118   }
6119
6120   \clist_set:No \l_tmpa_clist \spftype
6121   \tl_clear:N \l_tmpa_tl
6122   \clist_map_inline:Nn \l_tmpa_clist {
6123     \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
6124       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
6125     }
6126     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6127       \tl_set:Nn \l_tmpa_tl {\use:n{}}
6128     }
6129   }
6130   \tl_if_empty:NTF \l_tmpa_tl {
6131     \__stex_sproof_spfeq_start:
6132   }{
6133     \l_tmpa_tl
6134   }{~#2}
6135   \str_if_empty:NF \spfid {
6136     \stex_ref_new_doc_target:n \spfid
6137   }
6138   \begin{displaymath}\begin{array}{rcll}
6139 \end{array}
6140 \stex_smsmode_do:
6141 }{
6142   \stex_if_smsmode:F {
6143     \end{array}\end{displaymath}
6144     \clist_set:No \l_tmpa_clist \spftype
6145     \tl_clear:N \l_tmpa_tl
6146     \clist_map_inline:Nn \l_tmpa_clist {
6147       \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
6148         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
6149       }
6150     }
6151     \tl_if_empty:NTF \l_tmpa_tl {
6152       \__stex_sproof_spfeq_end:

```



```

6153     }{
6154         \l_tmpa_tl
6155     }
6156     \end{stex_annotate_env}
6157 }
6158 }
6159
6160 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
6161     \titleemph{
6162         \tl_if_empty:NTF \spftitle {
6163             \spf@proof@kw
6164         }{
6165             \spftitle
6166         }
6167     }:
6168 }
6169 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
6170
6171 \newcommand\stexpatchspfeq[3] [] {
6172     \str_set:Nx \l_tmpa_str{ #1 }
6173     \str_if_empty:NTF \l_tmpa_str {
6174         \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
6175         \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
6176     }{
6177         \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
6178         \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
6179     }
6180 }
6181

```

(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.

```

6182 \newenvironment{sproof}[2] []{
6183     \let \premise \stex_proof_premise:
6184     \intarray_gzero:N \l__stex_sproof_counter_intarray
6185     \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
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}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
6203 \str_if_empty:NF \spftype {
6204   \stex_annotate_invisible:nnn{type}{\spftype}{}
6205 }
6206
6207 \clist_set:No \l_tmpa_clist \spftype
6208 \tl_clear:N \l_tmpa_tl
6209 \clist_map_inline:Nn \l_tmpa_clist {
6210   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
6211     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
6212   }
6213   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6214     \tl_set:Nn \l_tmpa_tl {\use:n{}}
6215   }
6216 }
6217 \tl_if_empty:NTF \l_tmpa_tl {
6218   \__stex_sproof_sproof_start:
6219 }{
6220   \l_tmpa_tl
6221 }{~#2}
6222 \str_if_empty:NF \spfid {
6223   \stex_ref_new_doc_target:n \spfid
6224 }
6225 \begin{description}
6226 }
6227 \stex_smsmode_do:
6228 }{
6229   \stex_if_smsmode:F{
6230     \end{description}
6231     \clist_set:No \l_tmpa_clist \spftype
6232     \tl_clear:N \l_tmpa_tl
6233     \clist_map_inline:Nn \l_tmpa_clist {
6234       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
6235         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
6236       }
6237     }
6238     \tl_if_empty:NTF \l_tmpa_tl {
6239       \__stex_sproof_sproof_end:
6240     }{
6241       \l_tmpa_tl
6242     }
6243     \end{stex_annotate_env}
6244   }
6245 }
6246
6247 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6248   \par\noindent\titleemph{
6249     \tl_if_empty:NTF \spftype {
6250       \spf@proof@kw
6251     }{
6252       \spftype
6253     }
6254   }::

```

```

6255 }
6256 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6257
6258 \newcommand\stexpatchproof[3] [] {
6259   \str_set:Nx \l_tmpa_str{ #1 }
6260   \str_if_empty:NTF \l_tmpa_str {
6261     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6262     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6263   }{
6264     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6265     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6266   }
6267 }

```

\spfidea

```

6268 \newcommand\spfidea[2] []{
6269   \__stex_sproof_spf_args:n{#1}
6270   \titleemph{
6271     \tl_if_empty:NTF \spftype {Proof-Idea}{
6272       \spftype
6273     }:
6274   }~#2
6275   \sproofend
6276 }

```

(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

```

6277 \newenvironment{spfstep}[1] []{
6278   \__stex_sproof_spf_args:n{#1}
6279   \stex_if_smsmode:TF {
6280     \str_if_empty:NF \spfid {
6281       \stex_ref_new_doc_target:n \spfid
6282     }
6283   }{
6284     \@in@omtexttrue
6285     \clist_set:No \l_tmpa_clist \spftype
6286     \tl_set:Nn \l_tmpa_tl {
6287       \item[\sproofnumber]
6288       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6289     }
6290     \clist_map_inline:Nn \l_tmpa_clist {
6291       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6292         \tl_clear:N \l_tmpa_tl
6293       }
6294     }
6295     \l_tmpa_tl
6296     \seq_clear:N \l_tmpa_seq
6297     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {

```

```

6298     \tl_if_empty:nF{ ##1 }{
6299         \stex_get_symbol:n { ##1 }
6300         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6301             \l_stex_get_symbol_uri_str
6302         }
6303     }
6304 }
6305 \exp_args:Nnnx
6306 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6307 \str_if_empty:NF \spftype {
6308     \stex_annotate_invisible:nnn{type}{\spftype}{}}
6309 }
6310 \tl_if_empty:NF \spftitle {
6311     {(\titleemph{\spftitle})\enspace}
6312 }
6313 \str_if_empty:NF \spfid {
6314     \stex_ref_new_doc_target:n \spfid
6315 }
6316 }
6317 \stex_smsmode_do:
6318 \ignorespacesandpars
6319 }{
6320     \bool_if:NT \l__stex_sproof_inc_counter_bool {
6321         \__stex_sproof_inc_counter:
6322     }
6323     \stex_if_smsmode:F {
6324         \end{stex_annotate_env}
6325     }
6326 }

```

spfcomment

```

6327 \newenvironment{spfcomment}[1][]{
6328     \__stex_sproof_spf_args:n{#1}
6329     \clist_set:Nn \l_tmpa_clist \spftype
6330     \tl_set:Nn \l_tmpa_tl {
6331         \item[\sproofnumber]
6332         \bool_set_true:N \l__stex_sproof_inc_counter_bool
6333     }
6334     \clist_map_inline:Nn \l_tmpa_clist {
6335         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6336             \tl_clear:N \l_tmpa_tl
6337         }
6338     }
6339     \l_tmpa_tl
6340 }{
6341     \bool_if:NT \l__stex_sproof_inc_counter_bool {
6342         \__stex_sproof_inc_counter:
6343     }
6344 }

```

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.

```

6345 \newenvironment{subproof}[2][]{
6346   \_stex_sproof_spf_args:n{#1}
6347   \stex_if_smsmode:TF{
6348     \str_if_empty:NF \spfid {
6349       \stex_ref_new_doc_target:n \spfid
6350     }
6351   }{
6352     \seq_clear:N \l_tmpa_seq
6353     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6354       \tl_if_empty:nF{ ##1 }{
6355         \stex_get_symbol:n { ##1 }
6356         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6357           \l_stex_get_symbol_uri_str
6358         }
6359       }
6360     }
6361     \exp_args:Nnnx
6362     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {},}
6363     \str_if_empty:NF \spftype {
6364       \stex_annotate_invisible:nnn{type}{\spftype}{}
6365     }
6366
6367     \clist_set:No \l_tmpa_clist \spftype
6368     \tl_set:Nn \l_tmpa_tl {
6369       \item[\sproofnumber]
6370       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6371     }
6372     \clist_map_inline:Nn \l_tmpa_clist {
6373       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6374         \tl_clear:N \l_tmpa_tl
6375       }
6376     }
6377     \l_tmpa_tl
6378     \tl_if_empty:NF \spftitle {
6379       {(\titleemph{\spftitle})\enspace}
6380     }
6381     {\~#2}
6382     \str_if_empty:NF \spfid {
6383       \stex_ref_new_doc_target:n \spfid
6384     }
6385   }
6386   \_stex_sproof_add_counter:
6387   \stex_smsmode_do:
6388 }{
6389   \_stex_sproof_remove_counter:
6390   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6391     \_stex_sproof_inc_counter:
6392   }
6393   \stex_if_smsmode:F{
6394     \end{stex_annotate_env}
6395   }
6396 }

```

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

```

6397 \newenvironment{spfcases}[2] [] {
6398   \tl_if_empty:nTF{#1}{
6399     \begin{subproof}[method=by-cases]{#2}
6400   }{
6401     \begin{subproof}[#1,method=by-cases]{#2}
6402   }
6403 }{
6404   \end{subproof}
6405 }

```

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

```

6406 \newenvironment{spfcase}[2] [] {
6407   \__stex_sproof_spf_args:n{#1}
6408   \stex_if_smsmode:TF {
6409     \str_if_empty:NF \spfid {
6410       \stex_ref_new_doc_target:n \spfid
6411     }
6412   }{
6413     \seq_clear:N \l_tmpa_seq
6414     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6415       \tl_if_empty:nF{ ##1 }{
6416         \stex_get_symbol:n { ##1 }
6417         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6418           \l_stex_get_symbol_uri_str
6419         }
6420       }
6421     }
6422     \exp_args:Nnnx
6423     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6424     \str_if_empty:NF \spftype {
6425       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6426     }
6427     \clist_set:No \l_tmpa_clist \spftype
6428     \tl_set:Nn \l_tmpa_tl {
6429       \item[\sproofnumber]
6430       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6431     }
6432     \clist_map_inline:Nn \l_tmpa_clist {
6433       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6434         \tl_clear:N \l_tmpa_tl
6435       }
6436     }
6437     \l_tmpa_tl
6438     \tl_if_empty:nF{#2}{
6439       \titleemph{#2}:~
6440     }
6441   }
6442   \__stex_sproof_add_counter:
6443   \stex_smsmode_do:
6444 }{
6445   \__stex_sproof_remove_counter:
6446   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6447     \__stex_sproof_inc_counter:

```

```

6448 }
6449 \stex_if_smsmode:F{
6450   \clist_set:No \l_tmpa_clist \spftype
6451   \tl_set:Nn \l_tmpa_tl{\sproofend}
6452   \clist_map_inline:Nn \l_tmpa_clist {
6453     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6454       \tl_clear:N \l_tmpa_tl
6455     }
6456   }
6457   \l_tmpa_tl
6458   \end{stex_annotate_env}
6459 }
6460 }

```

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

```

6461 \newcommand\spfcasesketch[3] [] {
6462   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6463 }

```

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.

```

6464 \keys_define:nn { stex / just }{
6465   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6466   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6467   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6468   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6469 }

```

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

\spfjust

```

6470 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6471 \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.

```

6472 \newcommand\justarg[2] [] {#2}
6473 \</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

```
6474 <*package>
6475
6476 %%%%%%%%%% others.dtx %%%%%%%%%%
6477
6478 <@@=stex_others>
        Warnings and error messages
6479 % None

\MSC Math subject classifier

6480 \NewDocumentCommand \MSC {m} {
6481 % TODO
6482 }

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

6483 \@ifpackageloaded{tikzinput}{
6484 \RequirePackage{stex-tikzinput}
6485 }{}
6486
6487 \bool_if:NT \c_stex_persist_mode_bool {
6488 \let__stex_notation_restore_notation_old:nnnnn
6489 \__stex_notation_restore_notation:nnnnn
6490 \def__stex_notation_restore_notation_new:nnnnn#1#2#3#4#5{
6491 \__stex_notation_restore_notation_old:nnnnn{#1}{#2}{#3}{#4}{#5}
6492 \ExplSyntaxOn
6493 }
6494 \def__stex_notation_restore_notation:nnnnn{
6495 \ExplSyntaxOff
6496 \catcode'\sim10
6497 \__stex_notation_restore_notation_new:nnnnn
6498 }
6499 \input{\jobname.sms}
6500 \let__stex_notation_restore_notation:nnnnn
6501 \__stex_notation_restore_notation_old:nnnnn
6502 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
```



```

6503     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6504     \l_tmpa_str
6505     \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6506     \c_stex_mathhub_main_manifest_prop
6507     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6508   }
6509 }
6510 </package>

```

Chapter 35

STEX -Metatheory Implementation

```
6511 <*package>
6512 <@@=stex_modules>
6513
6514 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6515
6516 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6517 \begingroup
6518 \stex_module_setup:nn{
6519   ns=\c_stex_metatheory_ns_str,
6520   meta=NONE
6521 }{Metatheory}
6522 \stex_reactivate_macro:N \symdecl
6523 \stex_reactivate_macro:N \notation
6524 \stex_reactivate_macro:N \symdef
6525 \ExplSyntaxOff
6526 \csname stex_suppress_html:n\endcsname{
6527   % is-a (a:A, a \in A, a is an A, etc.)
6528   \symdecl{isa}[args=ai]
6529   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6530   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6531   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6532
6533   % bind (\forall, \Pi, \lambda etc.)
6534   \symdecl{bind}[args=Bi,assoc=pre]
6535   \notation{bind}[depfun,prec=nobrackets,op={(\cdot)\;\to\;\cdot}]{\comp( #1 \comp{}\;\to\;)}
6536   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6537   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6538
6539   % implicit bind
6540   \symdecl{implicitbind}[args=Bi,assoc=pre]
6541   \notation{implicitbind}[braces,prec=nobrackets,op={(\cdot\_I\;\cdot)}]{\comp\{ #1 \comp{
6542     \notation{implicitbind}[depfun,prec=nobrackets]{\comp( #1 \comp{}\;\to\_I\; } #2}{##1 \comp,
6543     \notation{implicitbind}[Pi]{\comp\prod^I_{#1}#2}{##1\comp,##2}
6544
6545   % dummy variable
```

```

6546 \symdecl{dummyvar}
6547 \notation{dummyvar}[underscore]{\comp\_}
6548 \notation{dummyvar}[dot]{\comp\cdot}
6549 \notation{dummyvar}[dash]{\comp{\rm --}}
6550
6551 %fromto (function space, Hom-set, implication etc.)
6552 \symdecl{fromto}[args=ai]
6553 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6554 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6555
6556 % mapto (lambda etc.)
6557 \symdecl{mapto}[args=Bi]
6558 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6559 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6560 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6561
6562 % function/operator application
6563 \symdecl{apply}[args=ia]
6564 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6565 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6566
6567 % collection of propositions/booleans/truth values
6568 \symdecl{prop}[name=proposition]
6569 \notation{prop}[prop]{\comp{\rm prop}}
6570 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6571
6572 \symdecl{judgmentholds}[args=1]
6573 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6574
6575 % sequences
6576 \symdecl{seqtype}[args=1]
6577 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6578
6579 \symdecl{seqexpr}[args=a]
6580 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6581
6582 \symdef{seqmap}[args=abi,setlike]{\comp\{#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6583 \symdef{seqprepend}[args=ia]{#1 \comp{::} #2}{##1 \comp, ##2}
6584 \symdef{seqappend}[args=ai]{#1 \comp{::} #2}{##1 \comp, ##2}
6585 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}}{##1\comp,##2}
6586 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}}{##1\comp,##2}
6587 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6588 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6589 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6590 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6591
6592 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6593 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{\comp{#2}}}
6594
6595 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses\,}}{##1\comp,##2}
6596 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses\,}#2}{##1\comp,##2}
6597 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{##1\comp,##2,##3}
6598
6599 % nat literals

```

```

6600 \symdef{natliteral}{\comp{\mathtt{Ord}}}
6601
6602 % letin (''let'', local definitions, variable substitution)
6603 \symdecl{letin}[args=bii]
6604 \notation{letin}{let}{\comp{\rm let}}\;#1\comp{=}\;#2\;\comp{\rm in}}\;#3}
6605 \notation{letin}{subst}{#3 \comp[ #1 \comp/ #2 \comp]}
6606 \notation{letin}{frac}{#3 \comp[ \frac{#2}{#1} \comp]}
6607
6608 % structures
6609 \symdecl*{module-type}[args=1]
6610 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6611 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6612 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6613
6614 % objects
6615 \symdecl{object}
6616 \notation{object}{\comp{\mathtt{OBJECT}}}
6617
6618 }
6619
6620 % The following are abbreviations in the sTeX corpus that are left over from earlier
6621 % developments. They will eventually be phased out.
6622
6623 \ExplSyntaxOn
6624 \stex_add_to_current_module:n{
6625   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6626   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6627   \def\livar{\csname sequence-index\endcsname[li]}
6628   \def\uivar{\csname sequence-index\endcsname[ui]}
6629   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6630   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6631 }
6632 \__stex_modules_end_module:
6633 \endgroup
6634 \</package>

```

Chapter 36

Tikzinput Implementation

```
6635 <@@=tikzinput>
6636 <*package>
6637
6638 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6639
6640 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6641 \RequirePackage{l3keys2e}
6642
6643 \keys_define:nn { tikzinput } {
6644   image .bool_set:N = \c_tikzinput_image_bool,
6645   image .default:n = false ,
6646   unknown .code:n = {}
6647 }
6648
6649 \ProcessKeysOptions { tikzinput }
6650
6651 \bool_if:NTF \c_tikzinput_image_bool {
6652   \RequirePackage{graphicx}
6653
6654   \providecommand\usetikzlibrary[]{}
6655   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6656 }{
6657   \RequirePackage{tikz}
6658   \RequirePackage{standalone}
6659
6660   \newcommand \tikzinput [2] [] {
6661     \setkeys{Gin}{#1}
6662     \ifx \Gin@ewidth \Gin@exclamation
6663       \ifx \Gin@eheight \Gin@exclamation
6664         \input { #2 }
6665       \else
6666         \resizebox{!}{ \Gin@eheight }{
6667           \input { #2 }
6668         }
6669       \fi
6670     \else
6671       \ifx \Gin@eheight \Gin@exclamation
6672         \resizebox{ \Gin@ewidth }{!}{
```

```

6673         \input { #2 }
6674     }
6675     \else
6676         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6677             \input { #2 }
6678         }
6679     \fi
6680 \fi
6681 }
6682 }
6683
6684 \newcommand \ctikzinput [2] [] {
6685     \begin{center}
6686         \tikzinput [#1] {#2}
6687     \end{center}
6688 }
6689
6690 \@ifpackageloaded{stex}{
6691     \RequirePackage{stex-tikzinput}
6692 }{}
6693
6694 </package>
6695 <*stex>
6696
6697 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6698 \RequirePackage{stex}
6699 \RequirePackage{tikzinput}
6700
6701 \newcommand\mhtikzinput[2] []{%
6702     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6703     \stex_in_repository:nn\Gin@mhrepos{
6704         \tikzinput[#1]{\mhp@hpath{##1}{#2}}
6705     }
6706 }
6707
6708 \newcommand\cmhtikzinput[2] []{\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6709
6710 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6711     \pgfkeys@spdef\pgf@temp{#1}
6712     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6713     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6714     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6715     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6716     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6717     \catcode'\@=11
6718     \catcode'\|=12
6719     \catcode'\$=3
6720     \pgfutil@InputIfFileExists{#2}{-}{-}
6721     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6722     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6723     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6724 }
6725
6726 \newcommand\libusetikzlibrary[1]{

```

```

6726 \prop_if_exist:NF \l_stex_current_repository_prop {
6727   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6728 }
6729 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6730   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6731 }
6732 \seq_clear:N \l__tikzinput_libinput_files_seq
6733 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6734 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6735
6736 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6737   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6738   \IfFileExists{ \l_tmpa_str }{
6739     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6740   }{
6741     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6742     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6743   }
6744 }
6745 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6746 \IfFileExists{ \l_tmpa_str }{
6747   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6748 }{ }
6749
6750 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6751   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6752 }{
6753   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6754     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6755       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6756     }
6757   }{
6758     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6759   }
6760 }
6761 }
6762 </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

```
6763 <*package>
6764 <@@=document_structure>
6765 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6766 \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).

```
6767
6768 \keys_define:nn{ document-structure }{
6769   class      .str_set_x:N = \c_document_structure_class_str,
6770   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6771   unknown    .code:n      = {
6772     \PassOptionsToClass{\CurrentOption}{stex}
6773     \PassOptionsToClass{\CurrentOption}{tikzinput}
6774   }
6775   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6776 }
6777 \ProcessKeysOptions{ document-structure }
6778 \str_if_empty:NT \c_document_structure_class_str {
6779   \str_set:Nn \c_document_structure_class_str {article}
6780 }
6781 \str_if_empty:NT \c_document_structure_topsect_str {
6782   \str_set:Nn \c_document_structure_topsect_str {section}
6783 }
```

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

```
6784 \RequirePackage{xspace}
6785 \RequirePackage{comment}
6786 \RequirePackage{stex}
6787 \AddToHook{begindocument}{
```



```

6788 \ltx@ifpackageloaded{babel}{
6789   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6790   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6791     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6792   }
6793 }{}
6794 }

```

`\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.

```

6795 \int_new:N \l_document_structure_section_level_int
6796 \str_case:VnF \c_document_structure_topsect_str {
6797   {part}}{
6798     \int_set:Nn \l_document_structure_section_level_int {0}
6799   }
6800   {chapter}{
6801     \int_set:Nn \l_document_structure_section_level_int {1}
6802   }
6803 }{
6804   \str_case:VnF \c_document_structure_class_str {
6805     {book}{
6806       \int_set:Nn \l_document_structure_section_level_int {0}
6807     }
6808     {report}{
6809       \int_set:Nn \l_document_structure_section_level_int {0}
6810     }
6811   }{
6812     \int_set:Nn \l_document_structure_section_level_int {2}
6813   }
6814 }

```

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

```

6815 \def\current@section@level{document}%
6816 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\xspace}%
6817 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%

```

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

`\skipfragment`

```

6818 \cs_new_protected:Npn \skipfragment {

```

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

```

6819 \ifcase\l_document_structure_section_level_int
6820 \or\stepcounter{part}
6821 \or\stepcounter{chapter}
6822 \or\stepcounter{section}
6823 \or\stepcounter{subsection}
6824 \or\stepcounter{subsubsection}
6825 \or\stepcounter{paragraph}
6826 \or\stepcounter{subparagraph}
6827 \fi
6828 }

```

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

blindfragment

```

6829 \newcommand\at@begin@blindsfragment[1]{
6830 \newenvironment{blindfragment}
6831 {
6832 \int_incr:N\l_document_structure_section_level_int
6833 \at@begin@blindsfragment\l_document_structure_section_level_int
6834 }{}

```

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

```

6835 \newcommand\sfragment@nonum[2]{
6836 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6837 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6838 }

```

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

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

```

6839 \newcommand\sfragment@num[2]{
6840 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6841 \@nameuse{#1}{#2}
6842 }{
6843 \cs_if_exist:NTF\rdfmata@sectioning{
6844 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6845 }{
6846 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6847 }
6848 }
6849 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6850 }

```

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

sfragment

```

6851 \keys_define:nn { document-structure / sfragment }{
6852 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6853 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6854 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6855 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6856 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6857 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6858 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6859 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6860 intro        .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6861 imports      .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6862 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6863 }
6864 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6865   \str_clear:N \l__document_structure_sfragment_id_str
6866   \str_clear:N \l__document_structure_sfragment_date_str
6867   \clist_clear:N \l__document_structure_sfragment_creators_clist
6868   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6869   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6870   \tl_clear:N \l__document_structure_sfragment_type_tl
6871   \tl_clear:N \l__document_structure_sfragment_short_tl
6872   \tl_clear:N \l__document_structure_sfragment_display_tl
6873   \tl_clear:N \l__document_structure_sfragment_imports_tl
6874   \tl_clear:N \l__document_structure_sfragment_intro_tl
6875   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6876   \keys_set:nn { document-structure / sfragment } { #1 }
6877 }

```

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.

```

6878 \newif\if@mainmatter\@mainmattertrue
6879 \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.

```

6880 \keys_define:nn { document-structure / sectioning }{
6881   name      .str_set_x:N = \l__document_structure_sect_name_str ,
6882   ref       .str_set_x:N = \l__document_structure_sect_ref_str  ,
6883   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6884   clear     .default:n   = {true} ,
6885   num       .bool_set:N  = \l__document_structure_sect_num_bool  ,
6886   num       .default:n   = {true}
6887 }
6888 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6889   \str_clear:N \l__document_structure_sect_name_str
6890   \str_clear:N \l__document_structure_sect_ref_str
6891   \bool_set_false:N \l__document_structure_sect_clear_bool
6892   \bool_set_false:N \l__document_structure_sect_num_bool
6893   \keys_set:nn { document-structure / sectioning } { #1 }
6894 }
6895 \newcommand\omdoc@sectioning[3][]{
6896   \l__document_structure_sect_args:n {#1 }
6897   \let\omdoc@sect@name\l__document_structure_sect_name_str
6898   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6899   \if@mainmatter% numbering not overridden by frontmatter, etc.
6900     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6901     \sfragment@num{#2}{#3}
6902   }{
6903     \sfragment@nonum{#2}{#3}
6904   }
6905   \def\current@section@level{\omdoc@sect@name}
6906 \else
6907   \sfragment@nonum{#2}{#3}
6908 \fi
6909 }% 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.

```

6910 \newcommand\sfragment@redefine@addtocontents[1]{%
6911   %\edef\__document_structureimport{#1}%
6912   %\@for\@I:=\__document_structureimport\do{%
6913     %\edef\@path{\csname module@\@I @path\endcsname}%
6914     %\@ifundefined{tf@toc}\relax%
6915     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6916     %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6917     %\def\addcontentsline##1##2##3{%
6918       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6919     %\else% hyperref.sty not loaded
6920     %\def\addcontentsline##1##2##3{%
6921       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6922     %\fi
6923   }% hypreref.sty loaded?

```

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

```

6924 \newenvironment{sfragment}[2] []% keys, title
6925 {
6926   \__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.

```

6927   \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6928
6929   \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6930     \sfragment@redefine@addtocontents{
6931       %\@ifundefined{module@id}\used@modules%
6932       %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6933     }
6934   }

```

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

```

6935
6936   \stex_document_title:n { #2 }
6937
6938   \int_incr:N\l__document_structure_section_level_int
6939   \ifcase\l__document_structure_section_level_int
6940     \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6941     \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6942 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6943 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6944 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6945 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6946 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6947 \fi
6948 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6949 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6950 \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6951 }
6952 }% for customization
6953 {}

```

and finally, we localize the sections

```

6954 \newcommand\omdoc@part@kw{Part}
6955 \newcommand\omdoc@chapter@kw{Chapter}
6956 \newcommand\omdoc@section@kw{Section}
6957 \newcommand\omdoc@subsection@kw{Subsection}
6958 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6959 \newcommand\omdoc@paragraph@kw{paragraph}
6960 \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`

```

6961 \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).

```

6962 \cs_if_exist:NTF\frontmatter{
6963 \let\__document_structure_orig_frontmatter\frontmatter
6964 \let\frontmatter\relax
6965 }{
6966 \tl_set:Nn\__document_structure_orig_frontmatter{
6967 \clearpage
6968 \@mainmatterfalse
6969 \pagenumbering{roman}
6970 }
6971 }
6972 \cs_if_exist:NTF\backmatter{
6973 \let\__document_structure_orig_backmatter\backmatter
6974 \let\backmatter\relax
6975 }{
6976 \tl_set:Nn\__document_structure_orig_backmatter{
6977 \clearpage
6978 \@mainmatterfalse

```

```

6979     \pagenumbering{roman}
6980   }
6981 }

```

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.

```

6982 \newenvironment{frontmatter}{
6983   \_document_structure_orig_frontmatter
6984 }{
6985   \cs_if_exist:NTF\mainmatter{
6986     \mainmatter
6987   }{
6988     \clearpage
6989     \@mainmattertrue
6990     \pagenumbering{arabic}
6991   }
6992 }

```

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

```

6993 \newenvironment{backmatter}{
6994   \_document_structure_orig_backmatter
6995 }{
6996   \cs_if_exist:NTF\mainmatter{
6997     \mainmatter
6998   }{
6999     \clearpage
7000     \@mainmattertrue
7001     \pagenumbering{arabic}
7002   }
7003 }

```

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

```

7004 \@mainmattertrue\pagenumbering{arabic}

```

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

```

7005 \def \c__document_structure_document_str{document}
7006 \newcommand\afterprematurestop{}
7007 \def\prematurestop@endsfragment{
7008   \unless\ifx\@currenvir\c__document_structure_document_str
7009     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
7010       \expandafter\prematurestop@endsfragment
7011     }
7012   }
7013 \providecommand\prematurestop{
7014   \message{Stopping~sTeX~processing~prematurely}
7015   \prematurestop@endsfragment
7016   \afterprematurestop
7017   \end{document}
7018 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
7019 \RequirePackage{etoolbox}
7020 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
7021 \newrobustcmd\useSGvar[1]{%
7022   \@ifundefined{sTeX@Gvar@#1}
7023   {\PackageError{document-structure}
7024     {The sTeX Global variable #1 is undefined}
7025     {set it with \protect\setSGvar}}
7026   \@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.

```
7027 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
7028   \@ifundefined{sTeX@Gvar@#1}
7029   {\PackageError{document-structure}
7030     {The sTeX Global variable #1 is undefined}
7031     {set it with \protect\setSGvar}}
7032   {\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.

```
7033 \*cls)
7034 \@@=notesslides)
7035 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
7036 \RequirePackage{13keys2e}
7037
7038 \keys_define:nn{notesslides / cls}{
7039   class .str_set_x:N = \c__notesslides_class_str,
7040   notes .bool_set:N = \c__notesslides_notes_bool ,
7041   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
7042   docopt .str_set_x:N = \c__notesslides_docopt_str,
7043   unknown .code:n = {
7044     \PassOptionsToPackage{\CurrentOption}{document-structure}
7045     \PassOptionsToClass{\CurrentOption}{beamer}
7046     \PassOptionsToPackage{\CurrentOption}{notesslides}
7047     \PassOptionsToPackage{\CurrentOption}{stex}
7048   }
7049 }
7050 \ProcessKeysOptions{ notesslides / cls }
7051
7052 \str_if_empty:NF \c__notesslides_class_str {
7053   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
7054 }
7055
7056 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
7057   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7058 }
7059 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
7060   \PassOptionsToPackage{defaultttopsect=part}{notesslides}
7061 }
7062
7063 \RequirePackage{stex}
```



```

7064 \stex_html_backend:T {
7065   \bool_set_true:N\c__notesslides_notes_bool
7066 }
7067
7068 \bool_if:NTF \c__notesslides_notes_bool {
7069   \PassOptionsToPackage{notes=true}{notesslides}
7070   \message{notesslides.cls:~Formatting~course~materials~in~notes~mode}
7071 }{
7072   \PassOptionsToPackage{notes=false}{notesslides}
7073   \message{notesslides.cls:~Formatting~course~materials~in~slides~mode}
7074 }
7075 </cls>

```

now we do the same for the notesslides package.

```

7076 <*package>
7077 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
7078 \RequirePackage{l3keys2e}
7079
7080 \keys_define:nn{notesslides / pkg}{
7081   topsect          .str_set_x:N = \c__notesslides_topsect_str,
7082   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
7083   notes            .bool_set:N = \c__notesslides_notes_bool ,
7084   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
7085   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
7086   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
7087   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
7088   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
7089   unknown          .code:n      = {
7090     \PassOptionsToClass{\CurrentOption}{stex}
7091     \PassOptionsToClass{\CurrentOption}{tikzinput}
7092   }
7093 }
7094 \ProcessKeysOptions{ notesslides / pkg }
7095
7096 \RequirePackage{stex}
7097 \stex_html_backend:T {
7098   \bool_set_true:N\c__notesslides_notes_bool
7099 }
7100
7101 \newif\ifnotes
7102 \bool_if:NTF \c__notesslides_notes_bool {
7103   \notesttrue
7104 }{
7105   \notestfalse
7106 }
7107

```

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

```

7108 \str_if_empty:NTF \c__notesslides_topsect_str {
7109   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
7110 }{
7111   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
7112 }
7113 \PassOptionsToPackage{topsect=\__notesslides_topsect}{document-structure}

```

```
7114 </package>
```

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

```
7115 <*cls>
7116 \bool_if:NTF \c__notesslides_notes_bool {
7117   \str_if_empty:NT \c__notesslides_class_str {
7118     \str_set:Nn \c__notesslides_class_str {article}
7119   }
7120   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_dcopt_str]
7121     {\c__notesslides_class_str}
7122 }{
7123   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
7124   \newcounter{Item}
7125   \newcounter{paragraph}
7126   \newcounter{subparagraph}
7127   \newcounter{Hfootnote}
7128 }
7129 \RequirePackage{document-structure}
```

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

```
7130 \RequirePackage{notesslides}
7131 </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.

```
7132 <*package>
7133 \bool_if:NT \c__notesslides_notes_bool {
7134   \RequirePackage{a4wide}
7135   \RequirePackage{marginnote}
7136   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
7137   \RequirePackage{mdframed}
7138   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
7139   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
7140 }
7141 \RequirePackage{stex-tikzinput}
7142 \RequirePackage{etoolbox}
7143 \RequirePackage{amssymb}
7144 \RequirePackage{amsmath}
7145 \RequirePackage{comment}
7146 \RequirePackage{textcomp}
7147 \RequirePackage{url}
7148 \RequirePackage{graphicx}
7149 \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

```

7150 \bool_if:NT \c__notesslides_notes_bool {
7151   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
7152 }
7153
7154
7155 \NewDocumentCommand \libusetheme {O{} m} {
7156   \bool_if:NTF \c__notesslides_notes_bool {
7157     \libusepackage[#1]{beamernotestheme#2}
7158   }{
7159     \libusepackage[#1]{beamertheme#2}
7160   }
7161 }
7162
```

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

```

7163 \newcounter{slide}
7164 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
7165 \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.

```

7166 \bool_if:NTF \c__notesslides_notes_bool {
7167   \renewenvironment{note}{\ignorespaces}{}
7168 }{
7169   \excludecomment{note}
7170 }

```

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.

```

7171 \bool_if:NT \c__notesslides_notes_bool {
7172   \newlength{\slideframewidth}
7173   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

7174 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
7175   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
7176     \bool_set_true:N #1
7177   }{
7178     \bool_set_false:N #1
7179   }
7180 }
7181 \keys_define:nn{notesslides / frame}{
7182   label .str_set_x:N = \l__notesslides_frame_label_str,
7183   allowframebreaks .code:n = {
7184     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
7185   },

```

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

```

7186 allowdisplaybreaks .code:n      = {
7187   \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
7188 },
7189 fragile .code:n      = {
7190   \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
7191 },
7192 shrink .code:n      = {
7193   \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
7194 },
7195 squeeze .code:n      = {
7196   \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
7197 },
7198 t .code:n      = {
7199   \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
7200 },
7201 unknown .code:n      = {}
7202 }
7203 \cs_new_protected:Nn \__notesslides_frame_args:n {
7204   \str_clear:N \l__notesslides_frame_label_str
7205   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
7206   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
7207   \bool_set_true:N \l__notesslides_frame_fragile_bool
7208   \bool_set_true:N \l__notesslides_frame_shrink_bool
7209   \bool_set_true:N \l__notesslides_frame_squeeze_bool
7210   \bool_set_true:N \l__notesslides_frame_t_bool
7211   \keys_set:nn { notesslides / frame }{ #1 }
7212 }

```

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

```

7213 \renewenvironment{frame}[1][ ]{
7214   \__notesslides_frame_args:n{#1}
7215   \sffamily
7216   \stepcounter{slide}
7217   \def\@currentlabel{\theslide}
7218   \str_if_empty:NF \l__notesslides_frame_label_str {
7219     \label{\l__notesslides_frame_label_str}
7220   }

```

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

```

7221 \def\itemize@level{outer}
7222 \def\itemize@outer{outer}
7223 \def\itemize@inner{inner}
7224 \renewcommand\newpage{\addtocounter{framenum}{1}}
7225 %\newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
7226 \renewenvironment{itemize}{
7227   \ifx\itemize@level\itemize@outer
7228     \def\itemize@label{$\rhd$}
7229   \fi
7230   \ifx\itemize@level\itemize@inner
7231     \def\itemize@label{$\scriptstyle\rhd$}
7232   \fi
7233   \begin{list}
7234     {\itemize@label}
7235     {\setlength{\labelsep}{.3em}
7236      \setlength{\labelwidth}{.5em}

```

```

7237     \setlength{\leftmargin}{1.5em}
7238   }
7239   \edef\itemize@level{\itemize@inner}
7240 }{
7241   \end{list}
7242 }

```

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

```

7243   \stex_html_backend:TF {
7244     \begin{stex_annotate_env}{frame}{}\vbox\bgroup
7245       \mdf@patchamsthm
7246   }{
7247     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
7248   }
7249 }{
7250   \stex_html_backend:TF {
7251     \miko@slidelabel\egroup\end{stex_annotate_env}
7252   }{\medskip\miko@slidelabel\end{mdframed}}
7253 }

```

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

```

\frametitle
7254   \renewcommand{\frametitle}[1]{
7255     \stex_document_title:n { #1 }
7256     {\Large\bf\sf\color{blue}{#1}}\medskip
7257   }
7258 }

```

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

EdN:19 `\pause` 19

```

7259   \bool_if:NT \c__notesslides_notes_bool {
7260     \newcommand\pause{}
7261   }

```

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

`nparagraph`

```

7262   \bool_if:NTF \c__notesslides_notes_bool {
7263     \newenvironment{nparagraph}[1] []{\begin{sparagraph}[#1]}\end{sparagraph}}
7264   }{
7265     \excludecomment{nparagraph}
7266   }

```

`nfragment`

```

7267   \bool_if:NTF \c__notesslides_notes_bool {
7268     \newenvironment{nfragment}[2] []{\begin{sfragment}[#1]{#2}}\end{sfragment}}
7269   }{
7270     \excludecomment{nfragment}
7271   }

```

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

ndefinition

```
7272 \bool_if:NTF \c__notesslides_notes_bool {
7273   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
7274 }{
7275   \excludecomment{ndefinition}
7276 }
```

nassertion

```
7277 \bool_if:NTF \c__notesslides_notes_bool {
7278   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
7279 }{
7280   \excludecomment{nassertion}
7281 }
```

nsproof

```
7282 \bool_if:NTF \c__notesslides_notes_bool {
7283   \newenvironment{nproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
7284 }{
7285   \excludecomment{nproof}
7286 }
```

nexample

```
7287 \bool_if:NTF \c__notesslides_notes_bool {
7288   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
7289 }{
7290   \excludecomment{nexample}
7291 }
```

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

```
7292 \def\inputref@preskip{\smallskip}
7293 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
7294 \let\orig@inputref\inputref
7295 \def\inputref{\@ifstar\ninputref\orig@inputref}
7296 \newcommand\ninputref[2] [] {
7297   \bool_if:NT \c__notesslides_notes_bool {
7298     \orig@inputref[#1]{#2}
7299   }
7300 }
```

(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>}`.

```

7301 \newlength{\slideloheight}
7302
7303 \RequirePackage{graphicx}
7304
7305 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7306 \providecommand\mhgraphics[2][]{
7307   \def\Gin@mhrepos{\setkeys{Gin}{#1}}
7308   \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
7309 }
7310
7311
7312
7313 \bool_if:NTF \c__notesslides_notes_bool {
7314   \setlength{\slideloheight}{.4cm}
7315 }{
7316   \setlength{\slideloheight}{1cm}
7317 }
7318 \ifcsname slidelo\endcsname\else
7319   \newsavebox{\slidelo}
7320   \sbox{\slidelo}{\TeX}
7321 \fi
7322 \newrobustcmd{\setslidelo}[2][]{
7323   \tl_if_empty:nTF{#1}{
7324     \sbox{\slidelo}{\includegraphics[height=\slideloheight]{#2}}
7325   }{
7326     \sbox{\slidelo}{\mhgraphics[height=\slideloheight,mhrepos=#1]{#2}}
7327   }
7328 }

```

(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.

```

7329 \def\source{Michael Kohlhase}% customize locally
7330 \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.

```

7331 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
7332 \newsavebox{\cclogo}
7333 \sbox{\cclogo}{\includegraphics[height=\slideloheight]{stex-cc_somerights}}
7334 \newif\ifcchref\cchreffalse
7335 \AtBeginDocument{
7336   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
7337 }
7338 \def\licensing{
7339   \ifcchref
7340     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}

```

```

7341 \else
7342   {\usebox{\cclogo}}
7343 \fi
7344 }
7345 \newrobustcmd{\setlicensing}[2][]{
7346   \def\@url{\#1}
7347   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
7348   \ifx\@url\@empty
7349     \def\licensing{\usebox{\cclogo}}
7350   \else
7351     \def\licensing{
7352       \ifcchref
7353       \href{\#1}{\usebox{\cclogo}}
7354       \else
7355       {\usebox{\cclogo}}
7356     \fi
7357   }
7358 \fi
7359 }

```

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

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

```

7360 \newrobustcmd\miko@slidelabel{
7361   \vbox to \slidelogoheight{
7362     \vss\hbox to \slidewidth
7363     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7364   }
7365 }

```

(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.

```

7366 \def\Gin@mhrefpos{}
7367 \define@key{Gin}{mhrepos}{\def\Gin@mhrefpos{\#1}}
7368 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
7369 \newrobustcmd\frameimage[2][]{
7370   \stepcounter{slide}
7371   \bool_if:NT \c__notesslides_frameimages_bool {
7372     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
7373     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7374     \begin{center}
7375       \bool_if:NTF \c__notesslides_fiboxed_bool {
7376         \fbox{
7377           \ifx\Gin@ewidth\@empty
7378             \ifx\Gin@mhrefpos\@empty
7379               \mhgraphics[width=\slidewidth,\#1]{\#2}
7380             \else
7381               \mhgraphics[width=\slidewidth,\#1,mhrefpos=\Gin@mhrefpos]{\#2}

```

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


```

7382         \fi
7383     \else% Gin@ewidth empty
7384         \ifx\Gin@mhrepos\@empty
7385             \mhgraphics[#1]{#2}
7386         \else
7387             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7388         \fi
7389     \fi% Gin@ewidth empty
7390 }
7391 }{
7392     \ifx\Gin@ewidth\@empty
7393     \ifx\Gin@mhrepos\@empty
7394         \mhgraphics[width=\slidewidth,#1]{#2}
7395     \else
7396         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7397     \fi
7398     \ifx\Gin@mhrepos\@empty
7399         \mhgraphics[#1]{#2}
7400     \else
7401         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7402     \fi
7403     \fi% Gin@ewidth empty
7404 }
7405 \end{center}
7406 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7407 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7408 }
7409 } % 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.

```

7410 \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

```

7411 \AddToHook{begindocument}{
7412     \definecolor{green}{rgb}{0,.5,0}
7413     \definecolor{purple}{cmyk}{.3,1,0,.17}
7414 }

```

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`.

```

7415 % \def\STpresent#1{\textcolor{blue}{#1}}
7416 \def\defemph#1{\textcolor{magenta}{#1}}
7417 \def\symrefemph#1{\textcolor{cyan}{#1}}
7418 \def\compemph#1{\textcolor{blue}{#1}}
7419 \def\titleemph#1{\textcolor{blue}{#1}}
7420 \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.

```

7421 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7422 \def\smalltextwarning{
7423   \pgfuseimage{miko@small@dbend}
7424   \xspace
7425 }
7426 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7427 \newrobustcmd\textwarning{
7428   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7429   \xspace
7430 }
7431 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7432 \newrobustcmd\bigtextwarning{
7433   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7434   \xspace
7435 }
(End definition for \textwarning. This function is documented on page 55.)
7436 \newrobustcmd\putgraphicsat[3]{
7437   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7438 }
7439 \newrobustcmd\putat[2]{
7440   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7441 }

```

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`.

```

7442 \stex_html_backend:F {
7443   \bool_if:NT \c__notesslides_sectocframes_bool {
7444     \str_if_eq:VnTF \__notesslidesstopsect{part}{
7445       \newcounter{chapter}\counterwithin*{section}{chapter}
7446     }{
7447       \str_if_eq:VnT \__notesslidesstopsect{chapter}{
7448         \newcounter{chapter}\counterwithin*{section}{chapter}
7449       }
7450     }
7451   }
7452 }
\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
7453 \def\part@prefix{}
7454 \@ifpackageloaded{document-structure}{\{
7455   \str_case:VnF \__notesslidesstopsect {
7456     {part}{

```

```

7457     \int_set:Nn \l_document_structure_section_level_int {0}
7458     \def\thesection{\arabic{chapter}.\arabic{section}}
7459     \def\part@prefix{\arabic{chapter}.}
7460   }
7461   {chapter}{
7462     \int_set:Nn \l_document_structure_section_level_int {1}
7463     \def\thesection{\arabic{chapter}.\arabic{section}}
7464     \def\part@prefix{\arabic{chapter}.}
7465   }
7466 }{
7467   \int_set:Nn \l_document_structure_section_level_int {2}
7468   \def\part@prefix{}
7469 }
7470 }
7471
7472 \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`

```

7473 \renewenvironment{sfragment}[2][]{
7474   \__document_structure_sfragment_args:n { #1 }
7475   \int_incr:N \l_document_structure_section_level_int
7476   \bool_if:NT \c__notesslides_sectocframes_bool {
7477     \stepcounter{slide}
7478     \begin{frame}[noframenumbering]
7479     \vfill\Large\centering
7480     \red{
7481       \ifcase\l_document_structure_section_level_int\or
7482         \stepcounter{part}
7483         \def\__notesslideslabel{\omdoc@part@kw}~\Roman{part}}
7484         \def\currentsectionlevel{\omdoc@part@kw}
7485       \or
7486         \stepcounter{chapter}
7487         \def\__notesslideslabel{\omdoc@chapter@kw}~\arabic{chapter}}
7488         \def\currentsectionlevel{\omdoc@chapter@kw}
7489       \or
7490         \stepcounter{section}
7491         \def\__notesslideslabel{\part@prefix\arabic{section}}
7492         \def\currentsectionlevel{\omdoc@section@kw}
7493       \or
7494         \stepcounter{subsection}
7495         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7496         \def\currentsectionlevel{\omdoc@subsection@kw}
7497       \or
7498         \stepcounter{subsubsection}
7499         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
7500         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7501       \or
7502         \stepcounter{paragraph}
7503         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{paragraph}}
7504         \def\currentsectionlevel{\omdoc@paragraph@kw}

```

```

7505         \else
7506             \def\__notesslideslabel{}
7507             \def\currentsectionlevel{\omdoc@paragraph@kw}
7508             \fi% end ifcase
7509             \__notesslideslabel%\sref@label@id\__notesslideslabel
7510             \quad #2%
7511         }%
7512         \vfill%
7513         \end{frame}%
7514     }
7515     \str_if_empty:NF \l__document_structure_sfragment_id_str {
7516         \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7517     }
7518 }{}
7519 }

```

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

```

7520 \def\inserttheorembodyfont{\normalfont}
7521 %\bool_if:NF \c__notesslides_notes_bool {
7522 % \defbeamertemplate{theorem begin}{miko}
7523 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7524 % \ifx\inserttheoremaddition\empty\else\ (\inserttheoremaddition)\fi%
7525 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7526 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7527 % \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.

```

7528 % \expandafter\def\csname Parent2\endcsname{}
7529 %}
7530
7531 \AddToHook{begindocument}{ % this does not work for some reasons
7532     \setbeamertemplate{theorems}[ams style]
7533 }
7534 \bool_if:NT \c__notesslides_notes_bool {
7535     \renewenvironment{columns}[1][{}]{%
7536         \par\noindent%
7537         \begin{minipage}%
7538             \linewidth\centering\leavevmode%
7539     }{%
7540         \end{minipage}\par\noindent%
7541     }%
7542     \newsavebox\columnbox%
7543     \renewenvironment<>{column}[2][{}]{%
7544         \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7545     }{%
7546         \end{minipage}\end{lrbox}\usebox\columnbox%
7547     }%
7548 }
7549 \bool_if:NFT \c__notesslides_noproblems_bool {
7550     \newenvironment{problems}{}{}

```

```

7551 }{
7552   \excludacomment{problems}
7553 }

```

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.

```

7554 \gdef\printexcursions{}
7555 \newcommand\excursionref[2]{% label, text
7556   \bool_if:NT \c__notesslides_notes_bool {
7557     \begin{sparagraph}[title=Excursion]
7558       #2 \sref[fallback=the appendix]{#1}.
7559     \end{sparagraph}
7560   }
7561 }
7562 \newcommand\activate@excursion[2][]{
7563   \gappto\printexcursions{\inputref[#1]{#2}}
7564 }
7565 \newcommand\excursion[4][]{% repos, label, path, text
7566   \bool_if:NT \c__notesslides_notes_bool {
7567     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7568   }
7569 }

```

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

\excursiongroup

```

7570 \keys_define:nn{notesslides / excursiongroup }{
7571   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7572   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7573   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7574 }
7575 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7576   \tl_clear:N \l__notesslides_excursion_intro_tl
7577   \str_clear:N \l__notesslides_excursion_id_str
7578   \str_clear:N \l__notesslides_excursion_mhrepos_str
7579   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7580 }
7581 \newcommand\excursiongroup[1][]{
7582   \__notesslides_excursion_args:n{ #1 }
7583   \ifdefempty\printexcursions{}% only if there are excursions
7584   {\begin{note}
7585     \begin{sfragment}[#1]{Excursions}%
7586     \ifdefempty\l__notesslides_excursion_intro_tl}{
7587       \inputref[\l__notesslides_excursion_mhrepos_str]{
7588         \l__notesslides_excursion_intro_tl
7589       }
7590     }
7591     \printexcursions%
7592     \end{sfragment}
7593   \end{note}}

```

```

7594 }
7595 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7596 \endpackage

```

(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.

```
7597 <*package>
7598 <@@=problems>
7599 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7600 \RequirePackage{l3keys2e,stex}
7601
7602 \keys_define:nn { problem / pkg }{
7603   notes      .default:n    = { true },
7604   notes      .bool_set:N   = \c__problems_notes_bool,
7605   gnotes     .default:n    = { true },
7606   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7607   hints      .default:n    = { true },
7608   hints      .bool_set:N   = \c__problems_hints_bool,
7609   solutions  .default:n    = { true },
7610   solutions  .bool_set:N   = \c__problems_solutions_bool,
7611   pts        .default:n    = { true },
7612   pts        .bool_set:N   = \c__problems_pts_bool,
7613   min        .default:n    = { true },
7614   min        .bool_set:N   = \c__problems_min_bool,
7615   boxed      .default:n    = { true },
7616   boxed      .bool_set:N   = \c__problems_boxed_bool,
7617   unknown    .code:n       = {}
7618 }
7619 \newif\ifsolutions
7620
7621 \ProcessKeysOptions{ problem / pkg }
7622 \bool_if:NTF \c__problems_solutions_bool {
7623   \solutionstrue
7624 }{
7625   \solutionsfalse
7626 }
```

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

```
7627 \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.

```
7628 \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.

```
7629 \def\prob@problem@kw{Problem}
7630 \def\prob@solution@kw{Solution}
7631 \def\prob@hint@kw{Hint}
7632 \def\prob@note@kw{Note}
7633 \def\prob@gnote@kw{Grading}
7634 \def\prob@pt@kw{pt}
7635 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7636 \AddToHook{begindocument}{
7637   \ltx@ifpackageloaded{babel}{
7638     \makeatletter
7639     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7640     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7641       \input{problem-ngerman.ldf}
7642     }
7643     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7644       \input{problem-finnish.ldf}
7645     }
7646     \clist_if_in:NnT \l_tmpa_clist {french}{
7647       \input{problem-french.ldf}
7648     }
7649     \clist_if_in:NnT \l_tmpa_clist {russian}{
7650       \input{problem-russian.ldf}
7651     }
7652     \makeatother
7653   }{}
7654 }
```

39.2 Problems and Solutions

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

```
7655 \keys_define:nn{ problem / problem }{
7656   id      .str_set_x:N = \l__problems_prob_id_str,
7657   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7658   min     .tl_set:N    = \l__problems_prob_min_tl,
7659   title   .tl_set:N    = \l__problems_prob_title_tl,
7660   type    .tl_set:N    = \l__problems_prob_type_tl,
7661   imports .tl_set:N    = \l__problems_prob_imports_tl,
7662   name    .str_set_x:N = \l__problems_prob_name_str,
7663   refnum  .int_set:N    = \l__problems_prob_refnum_int
```



```

7664 }
7665 \cs_new_protected:Nn \__problems_prob_args:n {
7666   \str_clear:N \l__problems_prob_id_str
7667   \str_clear:N \l__problems_prob_name_str
7668   \tl_clear:N \l__problems_prob_pts_tl
7669   \tl_clear:N \l__problems_prob_min_tl
7670   \tl_clear:N \l__problems_prob_title_tl
7671   \tl_clear:N \l__problems_prob_type_tl
7672   \tl_clear:N \l__problems_prob_imports_tl
7673   \int_zero_new:N \l__problems_prob_refnum_int
7674   \keys_set:nn { problem / problem }{ #1 }
7675   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7676     \let\l__problems_prob_refnum_int\undefined
7677   }
7678 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7679 \newcounter{problem}[section]
7680 \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.

```

7681 \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

```

7682 \newcommand\prob@number{
7683   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7684     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7685   }{
7686     \int_if_exist:NTF \l__problems_prob_refnum_int {
7687       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7688     }{
7689       \prob@label\theproblem
7690     }
7691   }
7692 }

```

(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.

```

7693 \newcommand\prob@title[3]{%
7694   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7695     #2 \l__problems_inclprob_title_tl #3
7696   }{
7697     \tl_if_exist:NTF \l__problems_prob_title_tl {
7698       #2 \l__problems_prob_title_tl #3
7699     }{
7700       #1

```

```

7701     }
7702   }
7703 }

```

(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.

```

7704 \def\prob@heading{
7705   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7706   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7707 }

```

(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`

```

7708 \newenvironment{sproblem}[1][{}]{
7709   \__problems_prob_args:n{#1}%\sref@target%
7710   \@in@omtexttrue% we are in a statement (for inline definitions)
7711   \stepcounter{problem}\record@problem
7712   \def\current@section@level{\prob@problem@kw}
7713
7714   \str_if_empty:NT \l__problems_prob_name_str {
7715     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7716     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7717     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7718   }
7719
7720   \stex_if_do_html:T{
7721     \tl_if_empty:NF \l__problems_prob_title_tl {
7722       \exp_args:No \stex_document_title:n \l__problems_prob_title_tl
7723     }
7724   }
7725
7726   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7727
7728   \stex_reactivate_macro:N \STEXexport
7729   \stex_reactivate_macro:N \importmodule
7730   \stex_reactivate_macro:N \symdecl
7731   \stex_reactivate_macro:N \notation
7732   \stex_reactivate_macro:N \symdef
7733
7734   \stex_if_do_html:T{
7735     \begin{stex_annotate_env} {problem} {
7736       \l_stex_module_ns_str ? \l_stex_module_name_str
7737     }
7738
7739     \stex_annotate_invisible:nnn{header}{} {
7740       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}

```

```

7741     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
7742     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7743         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
7744     }
7745 }
7746 }
7747
7748 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7749
7750
7751 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7752     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7753 }{
7754     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7755 }
7756 \str_if_exist:NTF \l__problems_inclprob_id_str {
7757     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7758 }{
7759     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7760 }
7761
7762
7763 \stex_if_smsmode:F {
7764     \clist_set:No \l_tmpa_clist \sproblemtype
7765     \tl_clear:N \l_tmpa_tl
7766     \clist_map_inline:Nn \l_tmpa_clist {
7767         \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7768             \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7769         }
7770     }
7771     \tl_if_empty:NTF \l_tmpa_tl {
7772         \__problems_sproblem_start:
7773     }{
7774         \l_tmpa_tl
7775     }
7776 }
7777 \stex_ref_new_doc_target:n \sproblemid
7778 \stex_smsmode_do:
7779 }{
7780     \__stex_modules_end_module:
7781     \stex_if_smsmode:F{
7782         \clist_set:No \l_tmpa_clist \sproblemtype
7783         \tl_clear:N \l_tmpa_tl
7784         \clist_map_inline:Nn \l_tmpa_clist {
7785             \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7786                 \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7787             }
7788         }
7789         \tl_if_empty:NTF \l_tmpa_tl {
7790             \__problems_sproblem_end:
7791         }{
7792             \l_tmpa_tl
7793         }
7794     }

```

```

7795 \stex_if_do_html:T{
7796   \end{stex_annotate_env}
7797 }
7798
7799 \smallskip
7800 }
7801
7802 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7803
7804
7805
7806 \cs_new_protected:Nn \__problems_sproblem_start: {
7807   \par\noindent\textbf{\prob@heading\show@pts\show@min\\\ignorespacesandpars
7808 }
7809 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7810
7811 \newcommand\stexpatchproblem[3][] {
7812   \str_set:Nx \l_tmpa_str{ #1 }
7813   \str_if_empty:NTF \l_tmpa_str {
7814     \tl_set:Nn \__problems_sproblem_start: { #2 }
7815     \tl_set:Nn \__problems_sproblem_end: { #3 }
7816   }{
7817     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7818     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7819   }
7820 }
7821
7822
7823 \bool_if:NT \c__problems_boxed_bool {
7824   \surroundwithmdframed{problem}
7825 }

```

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

```

7826 \def\record@problem{
7827   \protected@write\@auxout{}
7828   {
7829     \string\@problem{\prob@number}
7830     {
7831       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7832         \l__problems_inclprob_pts_tl
7833       }{
7834         \l__problems_prob_pts_tl
7835       }
7836     }%
7837     {
7838       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7839         \l__problems_inclprob_min_tl
7840       }{
7841         \l__problems_prob_min_tl
7842       }
7843     }
7844   }
7845 }

```

(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).

```
7846 \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.

```
7847 \keys_define:nn { problem / solution }{
7848   id          .str_set_x:N = \l__problems_solution_id_str ,
7849   for         .tl_set:N   = \l__problems_solution_for_tl ,
7850   height      .dim_set:N  = \l__problems_solution_height_dim ,
7851   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7852   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7853   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7854 }
7855 \cs_new_protected:Nn \__problems_solution_args:n {
7856   \str_clear:N \l__problems_solution_id_str
7857   \tl_clear:N \l__problems_solution_for_tl
7858   \tl_clear:N \l__problems_solution_srccite_tl
7859   \clist_clear:N \l__problems_solution_creators_clist
7860   \clist_clear:N \l__problems_solution_contributors_clist
7861   \dim_zero:N \l__problems_solution_height_dim
7862   \keys_set:nn { problem / solution }{ #1 }
7863 }
```

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

```
7864 \newcommand\@startsolution[1][]{
7865   \__problems_solution_args:n { #1 }
7866   \@in@omtexttrue% we are in a statement.
7867   \bool_if:NF \c__problems_boxed_bool { \hrule }
7868   \smallskip\noindent
7869   {\textbf{\prob@solution@kw :}\enspace}
7870   \begin{small}
7871   \def\current@section@level{\prob@solution@kw}
7872   \ignorespacesandpars
7873 }
```

`\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.

```
7874 \box_new:N \l__problems_solution_box
7875 \newenvironment{solution}[1][]{
7876   \stex_html_backend:TF{
7877     \stex_if_do_html:T{
7878       \begin{stex_annotate_env}{solution}{}}
7879   }
7880   }{
7881     \setbox\l__problems_solution_box\vbox\bgroup
7882     \par\smallskip\hrule\smallskip
7883     \noindent\textbf{Solution:}~
7884   }
7885   }{
7886     \stex_html_backend:TF{
```

```

7887 \stex_if_do_html:T{
7888 \end{stex_annotate_env}
7889 }
7890 }{
7891 \smallskip\hrule
7892 \egroup
7893 \bool_if:NT \c__problems_solutions_bool {
7894 \box\l__problems_solution_box
7895 }
7896 }
7897 }
7898
7899 \newcommand\startsolutions{
7900 \bool_set_true:N \c__problems_solutions_bool
7901 % \specialcomment{solution}{\@startsolution}{
7902 % \bool_if:NF \c__problems_boxed_bool {
7903 % \hrule\medskip
7904 % }
7905 % \end{small}%
7906 % }
7907 % \bool_if:NT \c__problems_boxed_bool {
7908 % \surroundwithmdframed{solution}
7909 % }
7910 }

```

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

\stopsolutions

```

7911 \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.

```

7912 \ifsolutions
7913 \startsolutions
7914 \else
7915 \stopsolutions
7916 \fi

```

exnote

```

7917 \bool_if:NTF \c__problems_notes_bool {
7918 \newenvironment{exnote}[1][{}{
7919 \par\smallskip\hrule\smallskip
7920 \noindent\textbf{\prob@note@kw :~ }\small
7921 }{
7922 \smallskip\hrule
7923 }
7924 }{
7925 \excludecomment{exnote}
7926 }

```

hint

```

7927 \bool_if:NTF \c__problems_notes_bool {
7928 \newenvironment{hint}[1][{}{
7929 \par\smallskip\hrule\smallskip

```

```

7930 \noindent\textbf{\prob@hint@kw :~ }\small
7931 }{
7932 \smallskip\hrule
7933 }
7934 \newenvironment{exhint}[1][]{
7935 \par\smallskip\hrule\smallskip
7936 \noindent\textbf{\prob@hint@kw :~ }\small
7937 }{
7938 \smallskip\hrule
7939 }
7940 }{
7941 \excludecomment{hint}
7942 \excludecomment{exhint}
7943 }

```

gnote

```

7944 \bool_if:NTF \c__problems_notes_bool {
7945 \newenvironment{gnote}[1][]{
7946 \par\smallskip\hrule\smallskip
7947 \noindent\textbf{\prob@gnote@kw :~ }\small
7948 }{
7949 \smallskip\hrule
7950 }
7951 }{
7952 \excludecomment{gnote}
7953 }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7954 \newenvironment{mcb}{
7955 \begin{enumerate}
7956 }{
7957 \end{enumerate}
7958 }

```

we define the keys for the mcb macro

```

7959 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7960 \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7961 \bool_set_true:N #1
7962 }{
7963 \bool_set_false:N #1
7964 }
7965 }
7966 \keys_define:nn { problem / mcb }{
7967 id .str_set_x:N = \l__problems_mcc_id_str ,
7968 feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7969 T .default:n = { false } ,
7970 T .bool_set:N = \l__problems_mcc_t_bool ,
7971 F .default:n = { false } ,
7972 F .bool_set:N = \l__problems_mcc_f_bool ,

```

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

```

7973 Ttext      .tl_set:N      = \l__problems_mcc_Ttext_str ,
7974 Ftext      .tl_set:N      = \l__problems_mcc_Ftext_str
7975 }
7976 \cs_new_protected:Nn \l__problems_mcc_args:n {
7977   \str_clear:N \l__problems_mcc_id_str
7978   \tl_clear:N \l__problems_mcc_feedback_tl
7979   \bool_set_false:N \l__problems_mcc_t_bool
7980   \bool_set_false:N \l__problems_mcc_f_bool
7981   \tl_clear:N \l__problems_mcc_Ttext_tl
7982   \tl_clear:N \l__problems_mcc_Ftext_tl
7983   \str_clear:N \l__problems_mcc_id_str
7984   \keys_set:nn { problem / mcc }{ #1 }
7985 }

```

\mcc

```

7986 \def\mccTrueText{\textbf{(true)}~}}
7987 \def\mccFalseText{\textbf{(false)}~}}
7988 \newcommand\mcc[2][]{}
7989   \l__problems_mcc_args:n{ #1 }
7990   \item[{$\Box$}] #2
7991   \ifsolutions
7992     \\\
7993     \bool_if:NT \l__problems_mcc_t_bool {
7994       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7995     }
7996     \bool_if:NT \l__problems_mcc_f_bool {
7997       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7998     }
7999     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
8000       \emph{(\l__problems_mcc_feedback_tl)}
8001     }
8002   \fi
8003 } %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.

```

8004
8005 \keys_define:nn{ problem / inclproblem }{
8006   id      .str_set_x:N = \l__problems_inclprob_id_str,
8007   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
8008   min     .tl_set:N    = \l__problems_inclprob_min_tl,
8009   title   .tl_set:N    = \l__problems_inclprob_title_tl,
8010   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
8011   type    .tl_set:N    = \l__problems_inclprob_type_tl,
8012   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
8013 }
8014 \cs_new_protected:Nn \__problems_inclprob_args:n {
8015   \str_clear:N \l__problems_prob_id_str

```



```

8016 \tl_clear:N \l__problems_inclprob_pts_tl
8017 \tl_clear:N \l__problems_inclprob_min_tl
8018 \tl_clear:N \l__problems_inclprob_title_tl
8019 \tl_clear:N \l__problems_inclprob_type_tl
8020 \int_zero_new:N \l__problems_inclprob_refnum_int
8021 \str_clear:N \l__problems_inclprob_mhrepos_str
8022 \keys_set:nn { problem / inclproblem }{ #1 }
8023 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
8024   \let\l__problems_inclprob_pts_tl\undefined
8025 }
8026 \tl_if_empty:NT \l__problems_inclprob_min_tl {
8027   \let\l__problems_inclprob_min_tl\undefined
8028 }
8029 \tl_if_empty:NT \l__problems_inclprob_title_tl {
8030   \let\l__problems_inclprob_title_tl\undefined
8031 }
8032 \tl_if_empty:NT \l__problems_inclprob_type_tl {
8033   \let\l__problems_inclprob_type_tl\undefined
8034 }
8035 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
8036   \let\l__problems_inclprob_refnum_int\undefined
8037 }
8038 }
8039
8040 \cs_new_protected:Nn \__problems_inclprob_clear: {
8041   \let\l__problems_inclprob_id_str\undefined
8042   \let\l__problems_inclprob_pts_tl\undefined
8043   \let\l__problems_inclprob_min_tl\undefined
8044   \let\l__problems_inclprob_title_tl\undefined
8045   \let\l__problems_inclprob_type_tl\undefined
8046   \let\l__problems_inclprob_refnum_int\undefined
8047   \let\l__problems_inclprob_mhrepos_str\undefined
8048 }
8049 \__problems_inclprob_clear:
8050
8051 \newcommand\includeproblem[2][ ]{
8052   \__problems_inclprob_args:n{ #1 }
8053   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
8054     \stex_html_backend:TF {
8055       \str_clear:N \l_tmpa_str
8056       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
8057         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
8058       }
8059       \stex_annotate_invisible:nnn{includeproblem}{
8060         \l_tmpa_str / #2
8061       }{}
8062     }{
8063       \begingroup
8064         \inputreftrue
8065         \tl_if_empty:nTF{ ##1 }{
8066           \input{#2}
8067         }{
8068           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
8069         }

```

```

8070     \endgroup
8071   }
8072 }
8073 \__problems_inclprob_clear:
8074 }

```

(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.

```

8075 \AddToHook{enddocument}{
8076   \bool_if:NT \c__problems_pts_bool {
8077     \message{Total:~\arabic{pts}~points}
8078   }
8079   \bool_if:NT \c__problems_min_bool {
8080     \message{Total:~\arabic{min}~minutes}
8081   }
8082 }

```

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

```

8083 \def\pts#1{
8084   \bool_if:NT \c__problems_pts_bool {
8085     \marginpar{#1~\prob@pt@kw}
8086   }
8087 }
8088 \def\min#1{
8089   \bool_if:NT \c__problems_min_bool {
8090     \marginpar{#1~\prob@min@kw}
8091   }
8092 }

```

`\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.

```

8093 \newcounter{pts}
8094 \def\show@pts{
8095   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
8096     \bool_if:NT \c__problems_pts_bool {
8097       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
8098       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
8099     }
8100   }{
8101     \tl_if_exist:NT \l__problems_prob_pts_tl {
8102       \bool_if:NT \c__problems_pts_bool {
8103         \tl_if_empty:NTF \l__problems_prob_pts_tl{
8104           \tl_set:Nn \l__problems_prob_pts_tl {0}
8105         }
8106         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
8107         \addtocounter{pts}{\l__problems_prob_pts_tl}
8108       }
8109     }

```

```

8110 }
8111 }

```

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

\show@min

```

8112 \newcounter{min}
8113 \def\show@min{
8114   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
8115     \bool_if:NT \c__problems_min_bool {
8116       \marginpar{\l__problems_inclprob_pts_tl\ min}
8117       \addtocounter{min}{\l__problems_inclprob_min_tl}
8118     }
8119   }{
8120     \tl_if_exist:NT \l__problems_prob_min_tl {
8121       \bool_if:NT \c__problems_min_bool {
8122         \tl_if_empty:NT\l__problems_prob_min_tl{
8123           \tl_set:Nn \l__problems_prob_min_tl {0}
8124         }
8125         \marginpar{\l__problems_prob_min_tl\ min}
8126         \addtocounter{min}{\l__problems_prob_min_tl}
8127       }
8128     }
8129   }
8130 }
8131 \</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.

```
8132 \*package>
8133 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
8134 \RequirePackage{13keys2e}
8135
8136 \newif\iftest\testfalse
8137 \DeclareOption{test}{\testtrue}
8138 \newif\ifmultiple\multiplefalse
8139 \DeclareOption{multiple}{\multipletrue}
8140 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
8141 \ProcessOptions
```

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

```
8142 \RequirePackage{keyval}[1997/11/10]
8143 \RequirePackage{problem}
```

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

```
8144 \newcommand\hwexam@assignment@kw{Assignment}
8145 \newcommand\hwexam@given@kw{Given}
8146 \newcommand\hwexam@due@kw{Due}
8147 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
8148 blank~for~extra~space}
8149 \def\hwexam@minutes@kw{minutes}
8150 \newcommand\correction@probs@kw{prob.}
8151 \newcommand\correction@pts@kw{total}
8152 \newcommand\correction@reached@kw{reached}
8153 \newcommand\correction@sum@kw{Sum}
8154 \newcommand\correction@grade@kw{grade}
8155 \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

```

8156 \AddToHook{begindocument}{
8157 \ltx@ifpackageloaded{babel}{
8158 \makeatletter
8159 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
8160 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
8161 \input{hwexam-ngerman.ldf}
8162 }
8163 \clist_if_in:NnT \l_tmpa_clist {finnish}{
8164 \input{hwexam-finnish.ldf}
8165 }
8166 \clist_if_in:NnT \l_tmpa_clist {french}{
8167 \input{hwexam-french.ldf}
8168 }
8169 \clist_if_in:NnT \l_tmpa_clist {russian}{
8170 \input{hwexam-russian.ldf}
8171 }
8172 \makeatother
8173 }{}
8174 }
8175

```

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.

```

8176 \newcounter{assignment}
8177 %\numberproblemsin{assignment}

```

We will prepare the keyval support for the `assignment` environment.

```

8178 \keys_define:nn { hwexam / assignment } {
8179 id .str_set:N = \l_@@_assign_id_str,
8180 number .int_set:N = \l_@@_assign_number_int,
8181 title .tl_set:N = \l_@@_assign_title_tl,
8182 type .tl_set:N = \l_@@_assign_type_tl,
8183 given .tl_set:N = \l_@@_assign_given_tl,
8184 due .tl_set:N = \l_@@_assign_due_tl,
8185 loadmodules .code:n = {
8186 \bool_set_true:N \l_@@_assign_loadmodules_bool
8187 }
8188 }
8189 \cs_new_protected:Nn \_@@_assignment_args:n {
8190 \str_clear:N \l_@@_assign_id_str
8191 \int_set:Nn \l_@@_assign_number_int {-1}
8192 \tl_clear:N \l_@@_assign_title_tl
8193 \tl_clear:N \l_@@_assign_type_tl
8194 \tl_clear:N \l_@@_assign_given_tl
8195 \tl_clear:N \l_@@_assign_due_tl
8196 \bool_set_false:N \l_@@_assign_loadmodules_bool
8197 \keys_set:nn { hwexam / assignment }{ #1 }
8198 }

```

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.

```

8199 \newcommand\given@due[2]{
8200 \bool_lazy_all:nF {
8201 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8202 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8203 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8204 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8205 }{ #1 }
8206
8207 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
8208 \tl_if_empty:NF \l_@@_assign_given_tl {
8209 \hwexam@given@kw\xspace\l_@@_assign_given_tl
8210 }
8211 }{
8212 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
8213 }
8214
8215 \bool_lazy_or:nnF {
8216 \bool_lazy_and_p:nn {
8217 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8218 }{
8219 \tl_if_empty_p:V \l_@@_assign_due_tl
8220 }
8221 }{
8222 \bool_lazy_and_p:nn {
8223 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
8224 }{
8225 \tl_if_empty_p:V \l_@@_assign_due_tl
8226 }
8227 }{ ,~ }
8228
8229 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
8230 \tl_if_empty:NF \l_@@_assign_due_tl {
8231 \hwexam@due@kw\xspace \l_@@_assign_due_tl
8232 }
8233 }{
8234 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
8235 }
8236
8237 \bool_lazy_all:nF {
8238 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
8239 { \tl_if_empty_p:V \l_@@_assign_given_tl }
8240 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
8241 { \tl_if_empty_p:V \l_@@_assign_due_tl }
8242 }{ #2 }
8243 }

```

`\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.

```

8244 \newcommand\assignment@title[3]{
8245 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
8246 \tl_if_empty:NTF \l_@@_assign_title_tl {
8247 #1
8248 }{
8249 #2\l_@@_assign_title_tl#3
8250 }
8251 }{
8252 #2\l_@@_inclasssign_title_tl#3
8253 }
8254 }

```

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

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

```

8255 \newcommand\assignment@number{
8256 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
8257 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8258 \arabic{assignment}
8259 } {
8260 \int_use:N \l_@@_assign_number_int
8261 }
8262 }{
8263 \int_use:N \l_@@_inclasssign_number_int
8264 }
8265 }

```

(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.

```

8266 \newenvironment{assignment}[1][ ]{
8267 \_@@_assignment_args:n { #1 }
8268 %\sref@target
8269 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
8270 \global\stepcounter{assignment}
8271 }{
8272 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
8273 }
8274 \setcounter{problem}{0}
8275 \renewcommand\prob@label[1]{\assignment@number.##1}
8276 \def\current@section@level{\document@hwexamtype}
8277 %\sref@label{id}{\document@hwexamtype \thesection}
8278 \begin{@assignment}
8279 }{
8280 \end{@assignment}
8281 }

```

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

```

8282 \def\ass@title{
8283 {\protect\document@hwexamtype}\arabic{assignment}
8284 \assignment@title{}\;\;{}{}\; -- \given@due{}\;}
8285 }
8286 \ifmultiple
8287 \newenvironment{@assignment}{
8288 \bool_if:NTF \l_@@_assign_loadmodules_bool {
8289 \begin{sfragment}[loadmodules]{\ass@title}
8290 }{
8291 \begin{sfragment}{\ass@title}
8292 }
8293 }{
8294 \end{sfragment}
8295 }

```

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

```

8296 \else
8297 \newenvironment{@assignment}{
8298 \begin{center}\bf
8299 \Large@title\strut\
8300 \document@hwexamtype}\arabic{assignment}\assignment@title{}\;\;{}{}\;
8301 \large\given@due{--\;\;{}{}\;}\;--}
8302 \end{center}
8303 }{}
8304 \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.

```

8305 \keys_define:nn { hwexam / inclassignment } {
8306 %id .str_set_x:N = \l_@@_assign_id_str,
8307 number .int_set:N = \l_@@_inclassign_number_int,
8308 title .tl_set:N = \l_@@_inclassign_title_tl,
8309 type .tl_set:N = \l_@@_inclassign_type_tl,
8310 given .tl_set:N = \l_@@_inclassign_given_tl,
8311 due .tl_set:N = \l_@@_inclassign_due_tl,
8312 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
8313 }
8314 \cs_new_protected:Nn \l_@@_inclassignment_args:n {
8315 \int_set:Nn \l_@@_inclassign_number_int {-1}
8316 \tl_clear:N \l_@@_inclassign_title_tl
8317 \tl_clear:N \l_@@_inclassign_type_tl
8318 \tl_clear:N \l_@@_inclassign_given_tl
8319 \tl_clear:N \l_@@_inclassign_due_tl
8320 \str_clear:N \l_@@_inclassign_mhrepos_str
8321 \keys_set:nn { hwexam / inclassignment }{ #1 }
8322 }
8323 \l_@@_inclassignment_args:n {}
8324
8325 \newcommand\inputassignment[2][]{

```



```

8326 \_@@_inclassassignment_args:n { #1 }
8327 \str_if_empty:NTF \l_@@_inclasssign_mhrepos_str {
8328 \input{#2}
8329 }{
8330 \stex_in_repository:nn{\l_@@_inclasssign_mhrepos_str}{
8331 \input{\mhpath{\l_@@_inclasssign_mhrepos_str}{#2}}
8332 }
8333 }
8334 \_@@_inclassassignment_args:n {}
8335 }
8336 \newcommand\includeassignment[2][ ]{
8337 \newpage
8338 \inputassignment[#1]{#2}
8339 }

```

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

40.4 Typesetting Exams

\quizheading

```

8340 \ExplSyntaxOff
8341 \newcommand\quizheading[1]{%
8342 \def\@tas{#1}%
8343 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
8344 \ifx\@tas\@empty\else%
8345 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
8346 \fi%
8347 }
8348 \ExplSyntaxOn

```

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

\testheading

```

8349
8350 \def\hwexamheader{\input{hwexam-default.header}}
8351
8352 \def\hwexamminutes{
8353 \tl_if_empty:NTF \testheading@duration {
8354 {\testheading@min}~\hwexam@minutes@kw
8355 }{
8356 \testheading@duration
8357 }
8358 }
8359
8360 \keys_define:nn { hwexam / testheading } {
8361 min .tl_set:N = \testheading@min,
8362 duration .tl_set:N = \testheading@duration,
8363 reqpts .tl_set:N = \testheading@reqpts,
8364 tools .tl_set:N = \testheading@tools
8365 }
8366 \cs_new_protected:Nn \_@@_testheading_args:n {
8367 \tl_clear:N \testheading@min
8368 \tl_clear:N \testheading@duration

```

```

8369 \tl_clear:N \testheading@reqpts
8370 \tl_clear:N \testheading@tools
8371 \keys_set:nn { hwexam / testheading }{ #1 }
8372 }
8373 \newenvironment{testheading}[1][]{
8374 \_@@_testheading_args:n{ #1 }
8375 \newcount\check@time\check@time=\testheading@min
8376 \advance\check@time by -\theassignment@totalmin
8377 \newif\if@bonuspoints
8378 \tl_if_empty:NTF \testheading@reqpts {
8379 \@bonuspointsfalse
8380 }{
8381 \newcount\bonus@pts
8382 \bonus@pts=\theassignment@totalpts
8383 \advance\bonus@pts by -\testheading@reqpts
8384 \edef\bonus@pts{\the\bonus@pts}
8385 \@bonuspointstrue
8386 }
8387 \edef\check@time{\the\check@time}
8388
8389 \makeatletter\hwexamheader\makeatother
8390 }{
8391 \newpage
8392 }

```

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

\testspace

```

8393 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8394 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8395 \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.

```

8396 <@@=problems>
8397 \renewcommand\@problem[3]{
8398 \stepcounter{assignment@probs}
8399 \def\__problemspts{#2}
8400 \ifx\__problemspts\@empty\else
8401 \addtocounter{assignment@totalpts}{#2}
8402 \fi
8403 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8404 \xdef\correction@probs{\correction@probs & #1}%
8405 \xdef\correction@pts{\correction@pts & #2}
8406 \xdef\correction@reached{\correction@reached &}

```

```

8407 }
8408 <@@=hwexam>

```

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

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

```

8409 \newcounter{assignment@probs}
8410 \newcounter{assignment@totalpts}
8411 \newcounter{assignment@totalmin}
8412 \def\correction@probs{\correction@probs@kw}
8413 \def\correction@pts{\correction@pts@kw}
8414 \def\correction@reached{\correction@reached@kw}
8415 \stepcounter{assignment@probs}
8416 \newcommand\correction@table{
8417 \resizebox{\textwidth}{!}{%
8418 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8419 &\multicolumn{\theassignment@probs}{c|}|%|
8420 {\footnotesize\correction@forgrading@kw} &\\ \hline
8421 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8422 \correction@pts & \theassignment@totalpts & \\ \hline
8423 \correction@reached & & \[.7cm]\hline
8424 \end{tabular}}
8425 </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).