

The sTeX3 Package Collection *

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

2022-04-25

Abstract

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

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

sTeX augments L^AT_EX with

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

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

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

*Version 3.0 (last revised 2022-04-25)

Contents

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

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

16	<code>sTeX</code>-Statements	82
16.1	Macros and Environments	82
17	<code>sTeX</code>-Proofs: Structural Markup for Proofs	83
18	<code>sTeX</code>-Metatheory	84
18.1	Symbols	84
III	Extensions	85
19	<code>Tikzinput</code>: Treating <code>TIKZ</code> code as images	86
19.1	Macros and Environments	86
20	<code>document-structure</code>: Semantic Markup for Open Mathematical Documents in <code>LaTeX</code>	87
21	<code>NotesSlides</code> – Slides and Course Notes	88
22	<code>problem.sty</code>: An Infrastructure for formatting Problems	89
23	<code>hwexam.sty/cls</code>: An Infrastructure for formatting Assignments and Exams	90
IV	Implementation	91
24	<code>sTeX</code>-Basics Implementation	92
24.1	The <code>sTeXDocument</code> Class	92
24.2	Preliminaries	92
24.3	Messages and logging	93
24.4	HTML Annotations	94
24.5	Babel Languages	95
24.6	Persistence	96
24.7	Auxiliary Methods	97
25	<code>sTeX</code>-MathHub Implementation	100
25.1	Generic Path Handling	100
25.2	PWD and <code>kpsewhich</code>	102
25.3	File Hooks and Tracking	103
25.4	MathHub Repositories	104
25.5	Using Content in Archives	109
26	<code>sTeX</code>-References Implementation	113
26.1	Document URIs and URLs	113
26.2	Setting Reference Targets	115
26.3	Using References	117
27	<code>sTeX</code>-Modules Implementation	120
27.1	The <code>smodule</code> environment	124
27.2	Invoking modules	130

28	sTeX-Module Inheritance Implementation	132
28.1	SMS Mode	132
28.2	Inheritance	136
29	sTeX-Symbols Implementation	141
29.1	Symbol Declarations	141
29.2	Notations	148
29.3	Variables	156
30	sTeX-Terms Implementation	163
30.1	Symbol Invocations	163
30.2	Terms	170
30.3	Notation Components	174
30.4	Variables	176
30.5	Sequences	178
31	sTeX-Structural Features Implementation	179
31.1	Imports with modification	180
31.2	The feature environment	188
31.3	Structure	188
32	sTeX-Statements Implementation	198
32.1	Definitions	198
32.2	Assertions	203
32.3	Examples	207
32.4	Logical Paragraphs	209
33	The Implementation	215
33.1	Proofs	215
33.2	Justifications	226
34	sTeX-Others Implementation	227
35	sTeX-Metatheory Implementation	228
36	Tikzinput Implementation	231
37	document-structure.sty Implementation	234
37.1	Package Options	234
37.2	Document Structure	235
37.3	Front and Backmatter	239
37.4	Global Variables	241
38	NotesSlides – Implementation	242
38.1	Class and Package Options	242
38.2	Notes and Slides	244
38.3	Header and Footer Lines	248
38.4	Frame Images	250
38.5	Colors and Highlighting	251
38.6	Sectioning	252
38.7	Excursions	254

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

Part I

Manual



Boxes like this one contain implementation details that are mostly relevant for more advanced use cases, might be useful to know when debugging, or might be good to know to better understand how something works. They can easily be skipped on a first read.



Boxes like this one explain how some $\text{\texttt{STeX}}$ concept relates to the MMT/OMDoc system, philosophy or language; see [MMT; Koh06] for introductions.

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

There are two ways of using 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 sTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

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

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

`\symname`

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

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

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

`\symref`

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

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

`\definame`
`\definiendum`

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

```

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

```

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

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

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

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

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

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

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

Remark 2.2.2:

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

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

Chapter 3

Creating sTeX Content

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

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

lang ($\langle\textit{language}\rangle*$) Languages to load with the `babel` package.

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

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

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

debug ($\langle\textit{log-prefix}\rangle*$) Logs debugging information with the given prefixes to the terminal, or all if `all` is given. Largely irrelevant for the majority of users.

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

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

$\hookrightarrow M \rightarrow$

$\hookrightarrow M \rightarrow$

$\hookrightarrow T \rightarrow$

- sTeX archives are simultaneously MMT archives, and the same directory structure is consequently used.

- sTeX modules correspond to OMDoc/MMT *theories*. `\importmodules` (and

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

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

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

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

3.2.1 The Local MathHub-Directory

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

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

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

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

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

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where \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{sparagraph}[type=symdoc,for=...]` environments for “encyclopaedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputrefing` snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.³

3.2.3 MANIFEST.MF-Files

The `MANIFEST.MF` in the `META-INF`-directory consists of key-value-pairs, informing \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/arithmetic,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

`source-base` or

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Remark 3.2.1:

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

```

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

```

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

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

Output:

Hello World

`\stexpatchmodule`

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

For example:

Example 2

Input:

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

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

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

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

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

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

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

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

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

Output:

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

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

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

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

$\backslash\text{setnotation}$

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

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

3.3.3 Argument Modes

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

Mode-b Arguments

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

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

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

Example 9

Input:

```

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

Output:

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

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

Mode-a Arguments

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

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

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

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

Example 10

Input:

```

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

```

Output:

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

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

Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

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

Mode-B Arguments

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

Example 12

Input:

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

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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

More precisely, each notation takes

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

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

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

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

In the example above:



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

3.3.6 Variables

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

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

`\svar`

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

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

`\vardef`

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

Example 19

Input:

```

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

```

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

`\varseq`

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

This is best shown by example:

Example 20

Input:


```

1 \vardef{varn}[name=n,type=\Nat]{\comp{n}}
2 \varseq{seqa}[name=a,type=\Nat]{1}{\varn}{\comp{a}_{#1}}
3
4 The  $i$ th index of  $\seqa!$  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 As expected, the above example is translated to OMDoc/MMT as an
 \hookrightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \hookrightarrow `<OMV name="m"/>` as arguments.



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

`\arg`

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

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

Example 35

Input:

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

Output:

Addition is...

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

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

Example 36

Input:

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

Output:

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

adding k yields...

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

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

Example 37

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

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

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 38

Input:

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

Output:

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

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

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

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

`\definiens`

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

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

Example 39

Input:

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

```

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

```

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

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

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

.

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

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

```

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

```

⁸EDNOTE: MK: reference

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

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

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

```

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

```

This yields the following result:

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

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

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

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

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

1.3. $n > 1$:

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

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

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

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

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

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

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

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

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

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

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

`\sproofend`

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

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

`\sProofEndSymbol`

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

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

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

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

```

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

```

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

```

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

```

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

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

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

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

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

```

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

```

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

Chapter 7

Additional Packages

7.1 Tikzinput: Treating TIKZ code as images

image

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

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

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

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

\tikzinput
\ctikzinput

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

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

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

`\mhtikzinput`
`\cmhtikzinput`

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

`\libusetikzlibrary`

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

7.2 Modular Document Structuring

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

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

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

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

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

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

```

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

¹⁰EdNOTE: MK: still?

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

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

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

```

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

```

Here we use two levels of `blindfragment`:

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

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

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

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

`\prematurestop`
`\afterprematurestop`

For prematurely stopping the formatting of a document, \TeX 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 \TeX preamble of the course notes file.

`\setSGvar`
`\useSGvar`

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

`\ifSGvar`

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

7.3 Slides and Course Notes

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

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

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

slides
notes
sectocframes
frameimages
fiboxed

The notesslides class takes a variety of class options:

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

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



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

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

```

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

```

`\ifnotes`

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

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



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



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

`\inputref*`

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

`nexample`, `nsproof`, `nassertion`

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

`\setslidelogo`

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

`\setsource`

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

`\setlicensing`

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

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

`\frameimage`
`\mhframeimage`

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

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

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

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

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

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

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

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

`\textwarning`

The `\textwarning` macro generates a warning sign: 

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

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

`\excursion`

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

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

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

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

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

`\excursiongroup`

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

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



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

7.4 Representing Problems and Solutions

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

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

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

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

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

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

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

Example 40

Input:

```

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

```

Output:

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

Hint: Think positively, this is simple!

Note: Justify your answer

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

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

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

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

\startsolutions
\stopsolutions

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

\ifsolutions

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

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

\mcc

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

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

If we start the solutions, then we get

Example 41

Input:

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

Output:

Problem 7.4.2 (Functions)

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

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

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

¹²EdNOTE: MK: that did not work!

Example 42

Input:

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

Output:

Problem 7.4.3 (Functions)

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

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

\includeproblem

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

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

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

7.5 Homeworks, Quizzes and Exams

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

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

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

```

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}

```

Will result in

Name:

Matriculation Number:

320101 General Computer Science (Fall 2010)

2022-04-25

You have one hour (sharp) for the test;

Write the solutions to the sheet.

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

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

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

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

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

good luck

EdN:13

13

\inputassignment

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

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

Part II

Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

<code>\if@latexml</code>	L ^A T _E X2e conditional for L ^A T _E X _{ML}
--------------------------	---

<code>\latexml_if_p: *</code>	L ^A T _E X3 conditionals for L ^A T _E X _{ML} .
<code>\latexml_if:TF *</code>	

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

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

We have four macros for annotating generated HTML (via L^AT_EX_{ML} or R_US_TE_X) with attributes:

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

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

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

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

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

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

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

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

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

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

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<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> <hr/> <code>\mhinput</code> <hr/>	<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> <hr/>	<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> <hr/>	<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> <hr/>	<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> <hr/> <code>\cmhgraphics</code> <hr/>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <hr/> <code>\clstinputmhlisting</code> <hr/>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 10

STEX-References

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

10.1 Macros and Environments

\STEXreftitle**\STEXreftitle{<some title>}**

Sets the title of the current document to *<some title>*. A reference to the current document from *some other* document will then be displayed accordingly. e.g. if **\STEXreftitle{foo book}** is called, then referencing Definition 3.5 in this document in another document will display **Definition 3.5 in foo book**.

\stex_get_document_uri:

Computes the current document uri from the current archive's **narr**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URI and the reference-id.

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

Computes the current URL from the current archive's **docurl**-field and its location relative to the archive's **source**-directory. Reference targets are computed from this URL and the reference-id, if this document is only included in SMS mode.

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n**\stex_ref_new_doc_target:n{<id>}**

Sets a new reference target with id *<id>*.

\stex_ref_new_sym_target:n**\stex_ref_new_sym_target:n{<uri>}**

Sets a new reference target for the symbol *<uri>*.

10.1.2 Using References

`\sref` `\sref[<opt-args>]{<id>}`

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

`\srefsym` `\srefsym[<opt-args>]{<symbol>}`

Like `\sref`, but references the *canonical label* for the provided symbol. The canonical target is the last of the following occurring in the document:

- A `\definiendum` or `\definame` for *<symbol>*,
- The `sassertion`, `sexample` or `sparagraph` with `for=<symbol>` that generated *<symbol>* in the first place, or
- A `\sparagraph` with `type=symdoc` and `for=<symbol>`.

`\srefsymuri` `\srefsymuri{<URI>}{<text>}`

A convenient short-hand for `\srefsym[linktext={<text>}]<URI>`, but requires the first argument to be a full URI already. Intended to be used in e.g. `\compemph@uri`, `\defemph@uri`, etc.

Chapter 11

STEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

The content of a module with uri $\langle URI \rangle$ is stored in four macros. All modifications of these macros are global:

`\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 {<archive-ID>} {<module-path>}`

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

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

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

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

- (b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

- (b) If `<path>` 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 {<ns>} {<archive-ID>} {<path>} {<name>}`

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTEX-Proofs: Structural Markup for Proofs

Chapter 18

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 </cls>
```

24.2 Preliminaries

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

```

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

```

\stex The sTeX logo:

\sTeX

```

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

```

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

24.3 Messages and logging

```

60 <@@=stex_log>

```

Warnings and error messages

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

95 <@@=stex_annotate>

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

96 \tl_new:N \l_stex_html_arg_tl

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

_stex_html_checkempty:n

```

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

```

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

\stex_if_do_html_p: Whether to (locally) produce HTML output

\stex_if_do_html:TF

```

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

```

```

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

```

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

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

```

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

```

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

`\stex_annotate:enw`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

```

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

```

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

24.5 Babel Languages

```

132 <@@=stex_language>

```

`\c_stex_languages_prop`

`\c_stex_language_abbrevs_prop`

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

```

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

```

```

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

```

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

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

```

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

```

24.6 Persistence

```

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

```

```

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

```

24.7 Auxiliary Methods

\stex_deactivate_macro:Nn

```

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

```

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

\stex_reactivate_macro:N

```

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

```

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

\ignorespacesandpars

```

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

```

```

229
230 \tl_clear:N \_tmp_args_tl
231 \int_step_inline:nn \l_tmpa_int {
232   \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}
233 }
234
235 \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}
236 \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{
237   \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
238   \exp_after:wN\exp_after:wN\exp_after:wN {
239     \exp_after:wN #2 \_tmp_args_tl
240   }
241 }}
242 }
243 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}
244 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}
245 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}

```

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

`\MMTrule`

```

246 \NewDocumentCommand \MMTrule {m m}{
247   \seq_set_split:Nnn \l_tmpa_seq , {#2}
248   \int_zero:N \l_tmpa_int
249   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
250     \seq_if_empty:NF \l_tmpa_seq {
251       $\seq_map_inline:Nn \l_tmpa_seq {
252         \int_incr:N \l_tmpa_int
253         \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
254       }$
255     }
256   }
257 }
258
259 \NewDocumentCommand \MMTinclude {m}{
260   \stex_annotate_invisible:nnn{import}{#1}{-}
261 }
262
263 \tl_new:N \g_stex_document_title
264 \cs_new_protected:Npn \STEXtitle #1 {
265   \tl_if_empty:NT \g_stex_document_title {
266     \tl_gset:Nn \g_stex_document_title { #1 }
267   }
268 }
269 \cs_new_protected:Nn \stex_document_title:n {
270   \tl_if_empty:NT \g_stex_document_title {
271     \tl_gset:Nn \g_stex_document_title { #1 }
272     \stex_annotate_invisible:nnn{doctitle}{-}{ #1 }
273   }
274 }
275 \AtBeginDocument {
276   \let \STEXtitle \stex_document_title:n
277   \tl_if_empty:NF \g_stex_document_title {
278     \stex_annotate_invisible:nnn{doctitle}{-}{ \g_stex_document_title }

```



```
279 }  
280 }  
281  
282 </package>
```

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

Chapter 25

STEX -MathHub Implementation

```
283 <*package>
284
285 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
286
287 <@@=stex_path>
288
289 Warnings and error messages
290 \msg_new:nnn{stex}{error/norepository}{
291   No~archive~#1~found~in~#2
292 }
293 \msg_new:nnn{stex}{error/notinarchive}{
294   Not~currently~in~an~archive,~but~\detokenize{#1}~
295   needs~one!
296 }
297 \msg_new:nnn{stex}{error/nofile}{
298   \detokenize{#1}~could~not~find~file~#2
299 }
300 \msg_new:nnn{stex}{error/twofiles}{
301   \detokenize{#1}~found~two~candidates~for~#2
302 }
```

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`

```
301 \cs_new_protected:Nn \stex_path_from_string:Nn {
302   \str_set:Nx \l_tmpa_str { #2 }
303   \str_if_empty:NTF \l_tmpa_str {
304     \seq_clear:N #1
305   }{
306     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
307     \sys_if_platform_windows:T{
308       \seq_clear:N \l_tmpa_tl
```

```

309     \seq_map_inline:Nn #1 {
310       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
311       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
312     }
313     \seq_set_eq:NN #1 \l_tmpa_tl
314   }
315   \stex_path_canonicalize:N #1
316 }
317 }
318

```

(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`

```

319 \cs_new_protected:Nn \stex_path_to_string:NN {
320   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
321 }
322
323 \cs_new:Nn \stex_path_to_string:N {
324   \seq_use:Nn #1 /
325 }

```

(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`

```

326 \str_const:Nn \c__stex_path_dot_str {.}
327 \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.

```

328 \cs_new_protected:Nn \stex_path_canonicalize:N {
329   \seq_if_empty:NF #1 {
330     \seq_clear:N \l_tmpa_seq
331     \seq_get_left:NN #1 \l_tmpa_tl
332     \str_if_empty:NT \l_tmpa_tl {
333       \seq_put_right:Nn \l_tmpa_seq {}
334     }
335     \seq_map_inline:Nn #1 {
336       \str_set:Nn \l_tmpa_tl { ##1 }
337       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
338         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
339           \seq_if_empty:NNTF \l_tmpa_seq {
340             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
341               \c__stex_path_up_str
342             }
343           }{
344             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
345             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
346               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
347                 \c__stex_path_up_str
348               }
349             }{

```

```

350         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
351     }
352 }
353 }{
354     \str_if_empty:NF \l_tmpa_tl {
355         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
356     }
357 }
358 }
359 }
360 \seq_gset_eq:NN #1 \l_tmpa_seq
361 }
362 }

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:NTF`

```

363 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
364     \seq_if_empty:NTF #1 {
365         \prg_return_false:
366     }{
367         \seq_get_left:NN #1 \l_tmpa_tl
368         \sys_if_platform_windows:TF{
369             \str_if_in:NnTF \l_tmpa_tl {:}{
370                 \prg_return_true:
371             }{
372                 \prg_return_false:
373             }
374         }{
375             \str_if_empty:NTF \l_tmpa_tl {
376                 \prg_return_true:
377             }{
378                 \prg_return_false:
379             }
380         }
381     }
382 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

383 \str_new:N\l_stex_kpsewhich_return_str
384 \cs_new_protected:Nn \stex_kpsewhich:n {
385     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
386     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
387     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
388 }

```

(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`

```

389 \sys_if_platform_windows:TF{
390   \begingroup\escapechar=-1\catcode'\=12
391   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
392   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
393   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
394   }}{
395   \stex_kpsewhich:n{-var-value~PWD}
396   }
397
398 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
399 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
400 \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

```

401 <@@=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

```

402 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

403 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
404 \stex_path_from_string:Nn \c_stex_mainfile_seq
405   \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`

```

406 \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`

```

407 \cs_new_protected:Nn \stex_filestack_push:n {
408   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
409   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
410     \stex_path_from_string:Nn\g_stex_currentfile_seq{
411       \c_stex_pwd_str/#1
412     }
413   }
414   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
415   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
416 }

```

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

`\stex_filestack_pop:`

```

417 \cs_new_protected:Nn \stex_filestack_pop: {
418   \seq_if_empty:NF\g__stex_files_stack{
419     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
420   }
421   \seq_if_empty:NTF\g__stex_files_stack{
422     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
423   }{
424     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
425     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
426   }
427 }
```

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

Hooks for the current file:

```

428 \AddToHook{file/before}{
429   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
430 }
431 \AddToHook{file/after}{
432   \stex_filestack_pop:
433 }
```

25.4 MathHub Repositories

434 `<@@=stex_mathhub>`

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

```

435 \str_if_empty:NTF\mathhub{
436   \sys_if_platform_windows:TF{
437     \begingroup\escapechar=-1\catcode'\=12
438     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
439     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
440     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
441   }{
442     \stex_kpsewhich:n{-var-value-MATHHUB}
443   }
444   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
445 }
446 \str_if_empty:NT \c_stex_mathhub_str {
447   \sys_if_platform_windows:TF{
448     \begingroup\escapechar=-1\catcode'\=12
449     \exp_args:Nx\stex_kpsewhich:n{-var-value-HOME}
450     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
451     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
452   }{
453     \stex_kpsewhich:n{-var-value-HOME}
454   }
455   \ior_open:NnT \l_tmpa_ior{\l_stex_kpsewhich_return_str / .stex / mathhub.path}{
456     \begingroup\escapechar=-1\catcode'\=12
457     \ior_str_get:NN \l_tmpa_ior \l_tmpa_str
```

```

458     \sys_if_platform_windows:T{
459       \exp_args:NNx\str_replace_all:Nnn\l_tmpa_str{\c_backslash_str}/
460     }
461     \str_gset_eq:NN \c_stex_mathhub_str\l_tmpa_str
462     \endgroup
463     \ior_close:N \l_tmpa_ior
464   }
465 }
466 \str_if_empty:NTF\c_stex_mathhub_str{
467   \msg_warning:nn{stex}{warning/nomathhub}
468 }{
469   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
470   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
471 }
472 }{
473   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
474   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
475     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
476       \c_stex_pwd_str/\mathhub
477     }
478   }
479   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
480   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
481 }

```

(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

```

482 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
483   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
484     \str_set:Nx \l_tmpa_str { #1 }
485     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
486     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
487     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
488     \_stex_mathhub_find_manifest:N \l_tmpa_seq
489     \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
490       \msg_error:nnxx{stex}{error/norepository}{#1}{
491         \stex_path_to_string:N \c_stex_mathhub_str
492       }
493     } {
494       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
495     }
496   }
497 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l_stex_mathhub_manifest_file_seq`

```

498 \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`:

```

499 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
500   \seq_set_eq:NN \l_tmpa_seq #1
501   \bool_set_true:N \l_tmpa_bool
502   \bool_while_do:Nn \l_tmpa_bool {
503     \seq_if_empty:NTF \l_tmpa_seq {
504       \bool_set_false:N \l_tmpa_bool
505     }{
506       \file_if_exist:nTF{
507         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
508       }{
509         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
510         \bool_set_false:N \l_tmpa_bool
511       }{
512         \file_if_exist:nTF{
513           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
514         }{
515           \seq_put_right:Nn \l_tmpa_seq{META-INF}
516           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
517           \bool_set_false:N \l_tmpa_bool
518         }{
519           \file_if_exist:nTF{
520             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
521           }{
522             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
523             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
524             \bool_set_false:N \l_tmpa_bool
525           }{
526             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
527           }
528         }
529       }
530     }
531   }
532   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
533 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

534 \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:

```

535 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
536   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
537   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
538   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
539     \str_set:Nn \l_tmpa_str {##1}
540     \exp_args:NNoo \seq_set_split:Nnn
541       \l_tmpb_seq \c_colon_str \l_tmpa_str
542     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```



```

543 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
544 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
545 }
546 \exp_args:No \str_case:nnTF \l_tmpa_tl {
547 {id} {
548 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
549 { id } \l_tmpb_tl
550 }
551 {narration-base} {
552 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
553 { narr } \l_tmpb_tl
554 }
555 {url-base} {
556 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
557 { docurl } \l_tmpb_tl
558 }
559 {source-base} {
560 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
561 { ns } \l_tmpb_tl
562 }
563 {ns} {
564 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
565 { ns } \l_tmpb_tl
566 }
567 {dependencies} {
568 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
569 { deps } \l_tmpb_tl
570 }
571 }{}{}
572 }{}
573 }
574 \ior_close:N \c__stex_mathhub_manifest_ior
575 \stex_persist:x {
576 \prop_set_from_keyval:cn{ c_stex_mathhub_#1_manifest_prop }{
577 \exp_after:wN \prop_to_keyval:N \csname c_stex_mathhub_#1_manifest_prop\endcsname
578 }
579 }
580 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

581 \cs_new_protected:Nn \stex_set_current_repository:n {
582 \stex_require_repository:n { #1 }
583 \prop_set_eq:Nc \l_stex_current_repository_prop {
584 c_stex_mathhub_#1_manifest_prop
585 }
586 }

```

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

`\stex_require_repository:n`

```

587 \cs_new_protected:Nn \stex_require_repository:n {
588 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
589 \stex_debug:nn{mathhub}{Opening~archive:~#1}

```

```

590   \_stex_mathhub_do_manifest:n { #1 }
591   }
592 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

593 %\prop_new:N \l_stex_current_repository_prop
594 \bool_if:NF \c_stex_persist_mode_bool {
595   \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
596   \seq_if_empty:NTF \l_stex_mathhub_manifest_file_seq {
597     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
598   } {
599     \_stex_mathhub_parse_manifest:n { main }
600     \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
601     \l_tmpa_str
602     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
603     \c_stex_mathhub_main_manifest_prop
604     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
605     \stex_debug:nn{mathhub}{Current~repository:~
606     \prop_item:Nn \l_stex_current_repository_prop {id}
607   }
608 }
609 }

```

(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.

```

610 \cs_new_protected:Nn \stex_in_repository:nn {
611   \str_set:Nx \l_tmpa_str { #1 }
612   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
613   \str_if_empty:NTF \l_tmpa_str {
614     \prop_if_exist:NTF \l_stex_current_repository_prop {
615       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
616       \exp_args:Ne \l_tmpa_cs{
617         \prop_item:Nn \l_stex_current_repository_prop { id }
618       }
619     }{
620       \l_tmpa_cs{}
621     }
622   }{
623     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
624     \stex_require_repository:n \l_tmpa_str
625     \str_set:Nx \l_tmpa_str { #1 }
626     \exp_args:Nne \use:nn {
627       \stex_set_current_repository:n \l_tmpa_str
628       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
629     }{
630       \stex_debug:nn{mathhub}{switching~back~to:~
631       \prop_if_exist:NTF \l_stex_current_repository_prop {
632         \prop_item:Nn \l_stex_current_repository_prop { id }::~
633       \meaning\l_stex_current_repository_prop
634     }{

```

```

635         no~repository
636     }
637 }
638 \prop_if_exist:NTF \l_stex_current_repository_prop {
639     \stex_set_current_repository:n {
640         \prop_item:Nn \l_stex_current_repository_prop { id }
641     }
642 }{
643     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
644 }
645 }
646 }
647 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

648 \def \mhpath #1 #2 {
649     \exp_args:Ne \tl_if_empty:nTF{#1}{
650         \c_stex_mathhub_str /
651         \prop_item:Nn \l_stex_current_repository_prop { id }
652         / source / #2
653     }{
654         \c_stex_mathhub_str / #1 / source / #2
655     }
656 }

```

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

`\inputref`

`\mhinput`

```

657 \newif \ifinputref \inputreffalse
658
659 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
660     \stex_in_repository:nn {#1} {
661         \ifinputref
662             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
663         \else
664             \inputreftrue
665             \input{ \c_stex_mathhub_str / ##1 / source / #2 }
666         \inputreffalse
667     \fi
668 }
669 }
670 \NewDocumentCommand \mhinput { 0{} m }{
671     \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
672 }
673
674 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
675     \stex_in_repository:nn {#1} {
676         \stex_html_backend:TF {
677             \str_clear:N \l_tmpa_str

```

```

678     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
679       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
680     }
681     \stex_annotate_invisible:nnn{inputref}{
682       \l_tmpa_str / #2
683     }{}
684   }{
685     \begingroup
686     \inputreftrue
687     \tl_if_empty:nTF{ ##1 }{
688       \input{#2}
689     }{
690       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
691     }
692     \endgroup
693   }
694 }
695 }
696 \NewDocumentCommand \inputref { 0{ } m }{
697   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
698 }

```

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

`\addmhbibresource`

```

699 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
700   \stex_in_repository:nn {#1} {
701     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
702   }
703 }
704 \newcommand\addmhbibresource[2][]{
705   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
706 }

```

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

`\libinput`

```

707 \cs_new_protected:Npn \libinput #1 {
708   \prop_if_exist:NF \l_stex_current_repository_prop {
709     \msg_error:nnn{stex}{error/notinarchive}\libinput
710   }
711   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
712     \msg_error:nnn{stex}{error/notinarchive}\libinput
713   }
714   \seq_clear:N \l__stex_mathhub_libinput_files_seq
715   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
716   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
717
718   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
719     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
720     \IfFileExists{ \l_tmpa_str }{
721       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
722     }{}
723     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
724     \seq_put_right:No \l_tmpa_seq \l_tmpa_str

```

```

725 }
726
727 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
728 \IfFileExists{ \l_tmpa_str }{
729   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
730 }{}
731
732 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
733   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
734 }{
735   \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
736     \input{ ##1 }
737   }
738 }
739 }

```

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

`\libusepackage`

```

740 \NewDocumentCommand \libusepackage {0{} m} {
741   \prop_if_exist:NF \l_stex_current_repository_prop {
742     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
743   }
744   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
745     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
746   }
747   \seq_clear:N \l__stex_mathhub_libinput_files_seq
748   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
749   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
750
751   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
752     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
753     \IfFileExists{ \l_tmpa_str.sty }{
754       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
755     }{}
756     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
757     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
758   }
759
760   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
761   \IfFileExists{ \l_tmpa_str.sty }{
762     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
763   }{}
764
765   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
766     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
767   }{
768     \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
769       \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
770         \usepackage[#1]{ ##1 }
771       }
772     }{
773       \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
774     }

```

```

775 }
776 }

```

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

```

\mhgraphics
\cmhgraphics

```

```

777
778 \AddToHook{begindocument}{
779 \ltx@ifpackageloaded{graphicx}{
780   \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
781   \newcommand\mhgraphics[2][]{\%
782     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
783     \includegraphics[#1]{\mhp@th\Gin@mhrepos{#2}}}
784   \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
785 }{}

```

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

```

\lstinputmhlisting
\clstinputmhlisting

```

```

786 \ltx@ifpackageloaded{listings}{
787   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
788   \newcommand\lstinputmhlisting[2][]{\%
789     \def\lst@mhrepos{}\setkeys{lst}{#1}%
790     \lstinputlisting[#1]{\mhp@th\lst@mhrepos{#2}}}
791   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
792 }{}
793 }
794
795 \</package>

```

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

Chapter 26

STEX -References Implementation

```
796 <*package>
797
798 %%%%%%%%%% references.dtx %%%%%%%%%%
799
800 <@@=stex_refs>
      Warnings and error messages
801
```

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

```
802 %\iow_new:N \c__stex_refs_refs_iow
803 \AtBeginDocument{
804 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
805 }
806 \AtEndDocument{
807 % \iow_close:N \c__stex_refs_refs_iow
808 }
```

`\STEXreftitle`

```
809 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
810
811 \NewDocumentCommand \STEXreftitle { m } {
812 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
813 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
814 \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:`

```
815 \cs_new_protected:Nn \stex_get_document_uri: {
816   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
817   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
818   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
819   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
820   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
821
822   \str_clear:N \l_tmpa_str
823   \prop_if_exist:NT \l_stex_current_repository_prop {
824     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
825       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
826     }
827   }
828
829   \str_if_empty:NTF \l_tmpa_str {
830     \str_set:Nx \l_stex_current_docns_str {
831       file:/\stex_path_to_string:N \l_tmpa_seq
832     }
833   }{
834     \bool_set_true:N \l_tmpa_bool
835     \bool_while_do:Nn \l_tmpa_bool {
836       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
837       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
838         {source} { \bool_set_false:N \l_tmpa_bool }
839       }{}{
840         \seq_if_empty:NT \l_tmpa_seq {
841           \bool_set_false:N \l_tmpa_bool
842         }
843       }
844     }
845
846     \seq_if_empty:NTF \l_tmpa_seq {
847       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
848     }{
849       \str_set:Nx \l_stex_current_docns_str {
850         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
851       }
852     }
853   }
854 }
```

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

`\l_stex_current_docurl_str`

```
855 \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:`

```
856 \cs_new_protected:Nn \stex_get_document_url: {
857   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
858   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
859   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```



```

860 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
861 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
862
863 \str_clear:N \l_tmpa_str
864 \prop_if_exist:NT \l_stex_current_repository_prop {
865   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
866     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
867       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
868     }
869   }
870 }
871
872 \str_if_empty:NTF \l_tmpa_str {
873   \str_set:Nx \l_stex_current_docurl_str {
874     file:/\stex_path_to_string:N \l_tmpa_seq
875   }
876 }{
877   \bool_set_true:N \l_tmpa_bool
878   \bool_while_do:Nn \l_tmpa_bool {
879     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
880     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
881       {source} { \bool_set_false:N \l_tmpa_bool }
882     }{}{
883       \seq_if_empty:NT \l_tmpa_seq {
884         \bool_set_false:N \l_tmpa_bool
885       }
886     }
887   }
888
889   \seq_if_empty:NTF \l_tmpa_seq {
890     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
891   }{
892     \str_set:Nx \l_stex_current_docurl_str {
893       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
894     }
895   }
896 }
897 }

```

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

26.2 Setting Reference Targets

```

898 \str_const:Nn \c__stex_refs_url_str{URL}
899 \str_const:Nn \c__stex_refs_ref_str{REF}
900 \str_new:N \l__stex_refs_curr_label_str
901 % @currentlabel -> number
902 % @currentlabelname -> title
903 % @currentHref -> name.number <- id of some kind
904 % \theH# -> \arabic{section}
905 % \the# -> number
906 % \hyper@makecurrent{#}
907 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

908 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
909   \stex_get_document_uri:
910   \str_clear:N \l__stex_refs_curr_label_str
911   \str_set:Nx \l_tmpa_str { #1 }
912   \str_if_empty:NT \l_tmpa_str {
913     \int_incr:N \l__stex_refs_unnamed_counter_int
914     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
915   }
916   \str_set:Nx \l__stex_refs_curr_label_str {
917     \l_stex_current_docns_str?\l_tmpa_str
918   }
919   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
920     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
921   }
922   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
923     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
924   }
925   \stex_if_smsmode:TF {
926     \stex_get_document_url:
927     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
928     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
929   }{
930     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
931       \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
932       \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
933       \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
934     }
935   }

```

(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.

```

936 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
937   \str_set:Nn \l_tmpa_str {#1?#2}
938   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
939   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
940     \seq_new:c {g__stex_refs_labels_#2_seq}
941   }
942   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
943     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
944   }
945 }

```

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

```

946 \AtEndDocument{
947   \def\stexauxadddocref#1 #2 {}{}
948 }

```

`\stex_ref_new_sym_target:n`

```

949 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
950   \stex_if_smsmode:TF {
951     \str_if_exist:cF{sref_sym_#1_type}{
952       \stex_get_document_url:
953       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

954     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
955   }
956 }{
957   \str_if_empty:NF \l__stex_refs_curr_label_str {
958     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
959     \immediate\write\@auxout{
960       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
961         \l__stex_refs_curr_label_str
962       }
963     }
964   }
965 }
966 }

```

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

26.3 Using References

```

967 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

968
969 \keys_define:nn { stex / sref } {
970   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
971   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
972   pre           .tl_set:N = \l__stex_refs_pre_tl ,
973   post          .tl_set:N = \l__stex_refs_post_tl ,
974 }
975 \cs_new_protected:Nn \__stex_refs_args:n {
976   \tl_clear:N \l__stex_refs_linktext_tl
977   \tl_clear:N \l__stex_refs_fallback_tl
978   \tl_clear:N \l__stex_refs_pre_tl
979   \tl_clear:N \l__stex_refs_post_tl
980   \str_clear:N \l__stex_refs_repo_str
981   \keys_set:nn { stex / sref } { #1 }
982 }

```

The actual macro:

```

983 \NewDocumentCommand \sref { 0{} m}{
984   \__stex_refs_args:n { #1 }
985   \str_if_empty:NTF \l__stex_refs_indocument_str {
986     \str_set:Nx \l_tmpa_str { #2 }
987     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
988     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
989       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
990         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
991           \str_clear:N \l_tmpa_str
992         }
993       }{
994         \str_clear:N \l_tmpa_str
995       }
996     }{
997       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
998       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

999     \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1000   \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1001     \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1002     \str_clear:N \l_tmpa_str
1003     \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1004       \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1005         \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
1006       }{
1007         \seq_map_break:n {
1008           \str_set:Nn \l_tmpa_str { ##1 }
1009         }
1010       }
1011     }
1012   }{
1013     \str_clear:N \l_tmpa_str
1014   }
1015 }
1016 \str_if_empty:NTF \l_tmpa_str {
1017   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1018 }{
1019   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1020     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1021       \cs_if_exist:cTF{autoref}{
1022         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1023       }{
1024         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1025       }
1026     }{
1027       \ltx@ifpackageloaded{hyperref}{
1028         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1029       }{
1030         \l__stex_refs_linktext_tl
1031       }
1032     }
1033   }{
1034     \ltx@ifpackageloaded{hyperref}{
1035       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1036     }{
1037       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1038     }
1039   }
1040 }
1041 }{
1042   % TODO
1043 }
1044 }

```

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

`\srefsym`

```

1045 \NewDocumentCommand \srefsym { 0{} m }{
1046   \stex_get_symbol:n { #2 }
1047   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1048 }

```

```

1049
1050 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1051   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1052     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1053   }{
1054     \__stex_refs_args:n { #1 }
1055     \str_if_empty:NTF \l__stex_refs_indocument_str {
1056       \tl_if_exist:cTF{sref_sym_#2 _type}{
1057         % doc uri in \l_tmpb_str
1058         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1059         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1060           % reference
1061           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1062             \cs_if_exist:cTF{autoref}{
1063               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1064             }{
1065               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1066             }
1067           }{
1068             \ltx@ifpackageloaded{hyperref}{
1069               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1070             }{
1071               \l__stex_refs_linktext_tl
1072             }
1073           }
1074         }{
1075           % URL
1076           \ltx@ifpackageloaded{hyperref}{
1077             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1078           }{
1079             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1080           }
1081         }
1082       }{
1083         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1084       }
1085     }{
1086       % TODO
1087     }
1088   }
1089 }

```

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

\srefsymuri

```

1090 \cs_new_protected:Npn \srefsymuri #1 #2 {
1091   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1092 }

```

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

```

1093 </package>

```

Chapter 27

STEX -Modules Implementation

```
1094 <*package>
1095
1096 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1097
1098 <@@=stex_modules>
1099
1100     Warnings and error messages
1101 \msg_new:nnn{stex}{error/unknownmodule}{
1102     No~module~#1~found
1103 }
1104 \msg_new:nnn{stex}{error/syntax}{
1105     Syntax~error:~#1
1106 }
1107 \msg_new:nnn{stex}{error/siglanguage}{
1108     Module~#1~declares~signature~#2,~but~does~not~
1109     declare~its~language
1110 }
1111 \msg_new:nnn{stex}{warning/deprecated}{
1112     #1~is~deprecated;~please~use~#2~instead!
1113 }
1114 \msg_new:nnn{stex}{error/conflictingmodules}{
1115     Conflicting~imports~for~module~#1
1116 }
1117
1118 \l_stex_current_module_str The current module:
1119 \str_new:N \l_stex_current_module_str
1120
1121 (End definition for \l_stex_current_module_str. This variable is documented on page 71.)
1122
1123 \l_stex_all_modules_seq Stores all available modules
1124 \seq_new:N \l_stex_all_modules_seq
1125
1126 (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
1118 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1119   \str_if_empty:NTF \l_stex_current_module_str
1120   \prg_return_false: \prg_return_true:
1121 }

(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
1122 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1123   \prop_if_exist:cTF { c_stex_module_#1_prop }
1124   \prg_return_true: \prg_return_false:
1125 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1126 \cs_new_protected:Nn \stex_execute_in_module:n { \stex_if_in_module:T {
1127   \stex_add_to_current_module:n { #1 }
1128   \stex_do_up_to_module:n { #1 }
1129 }}
1130 \cs_generate_variant:Nn \stex_execute_in_module:n {x}
1131
1132 \cs_new_protected:Nn \stex_add_to_current_module:n {
1133   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1134 }
1135 \cs_generate_variant:Nn \stex_add_to_current_module:n {x}
1136 \cs_new_protected:Npn \STEXexport {
1137   \beginngroup
1138   \newlinechar=-1\relax
1139   \endlinechar=-1\relax
1140   %\catcode'\ = 9\relax
1141   \expandafter\endgroup\__stex_modules_export:n
1142 }
1143 \cs_new_protected:Nn \__stex_modules_export:n {
1144   \ignorespaces #1
1145   \stex_add_to_current_module:n { \ignorespaces #1 }
1146   \stex_smsmode_do:
1147 }
1148 \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
1149 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1150   \str_set:Nx \l_tmpa_str { #1 }
1151   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1152 }

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

```

`\stex_add_import_to_current_module:n`

```

1153 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1154   \str_set:Nx \l_tmpa_str { #1 }
1155   \exp_args:Nno
1156   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1157     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1158   }
1159 }

```

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

`\stex_collect_imports:n`

```

1160 \cs_new_protected:Nn \stex_collect_imports:n {
1161   \seq_clear:N \l_stex_collect_imports_seq
1162   \__stex_modules_collect_imports:n {#1}
1163 }
1164 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1165   \seq_map_inline:cn {c_stex_module_#1_imports} {
1166     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1167       \__stex_modules_collect_imports:n { ##1 }
1168     }
1169   }
1170   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1171     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1172   }
1173 }

```

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

`\stex_do_up_to_module:n`

```

1174 \int_new:N \l__stex_modules_group_depth_int
1175 \cs_new_protected:Nn \stex_do_up_to_module:n {
1176   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1177     #1
1178   }{
1179     #1
1180     \expandafter \tl_gset:Nn
1181     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1182     \expandafter\expandafter\expandafter\endcsname
1183     \expandafter\expandafter\expandafter { \csname
1184       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1185     \aftergroup\__stex_modules_aftergroup_do:
1186   }
1187 }
1188 \cs_generate_variant:Nn \stex_do_up_to_module:n {x}
1189 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1190   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1191     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1192   }}
1193   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1194     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1195     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1196   }{
1197     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}

```



```

1198     \aftergroup\__stex_modules_aftergroup_do:
1199   }
1200 }
1201 \cs_new_protected:Nn \_stex_reset_up_to_module:n {
1202   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1203 }

```

(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).

```

1204

```

(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.

```

1205 \str_new:N \l_stex_module_ns_str
1206 \str_new:N \l_stex_module_subpath_str
1207 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1208   \seq_set_eq:NN \l_tmpa_seq #2
1209   % split off file extension
1210   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1211   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1212   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1213   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1214
1215   \bool_set_true:N \l_tmpa_bool
1216   \bool_while_do:Nn \l_tmpa_bool {
1217     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1218     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1219       {source} { \bool_set_false:N \l_tmpa_bool }
1220     }{}{
1221       \seq_if_empty:NT \l_tmpa_seq {
1222         \bool_set_false:N \l_tmpa_bool
1223       }
1224     }
1225   }
1226
1227   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1228   % \l_tmpa_seq <- sub-path relative to archive
1229   \str_if_empty:NTF \l_stex_module_subpath_str {
1230     \str_set:Nx \l_stex_module_ns_str {#1}
1231   }{
1232     \str_set:Nx \l_stex_module_ns_str {
1233       #1/\l_stex_module_subpath_str
1234     }
1235   }
1236 }
1237
1238 \cs_new_protected:Nn \stex_modules_current_namespace: {
1239   \str_clear:N \l_stex_module_subpath_str
1240   \prop_if_exist:NTF \l_stex_current_repository_prop {
1241     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str

```

```

1242     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1243   }{
1244     % split off file extension
1245     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1246     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1247     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1248     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1249     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1250     \str_set:Nx \l_stex_module_ns_str {
1251       file:/\stex_path_to_string:N \l_tmpa_seq
1252     }
1253   }
1254 }

```

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

27.1 The smodule environment

smodule arguments:

```

1255 \keys_define:nn { stex / module } {
1256   title      .tl_set:N      = \smodulename ,
1257   type       .str_set_x:N   = \smodulename ,
1258   id         .str_set_x:N   = \smoduleid ,
1259   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1260   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1261   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1262   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1263   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1264   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1265   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1266   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1267 }
1268
1269 \cs_new_protected:Nn \__stex_modules_args:n {
1270   \str_clear:N \smodulename
1271   \str_clear:N \smoduleid
1272   \str_clear:N \l_stex_module_ns_str
1273   \str_clear:N \l_stex_module_deprecate_str
1274   \str_clear:N \l_stex_module_lang_str
1275   \str_clear:N \l_stex_module_sig_str
1276   \str_clear:N \l_stex_module_creators_str
1277   \str_clear:N \l_stex_module_contributors_str
1278   \str_clear:N \l_stex_module_meta_str
1279   \str_clear:N \l_stex_module_srccite_str
1280   \keys_set:nn { stex / module } { #1 }
1281 }
1282
1283 % module parameters here? In the body?
1284
1285

```

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

```

1286 \cs_new_protected:Nn \stex_module_setup:nn {

```

```

1287 \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1288 \str_set:Nx \l_stex_module_name_str { #2 }
1289 \__stex_modules_args:n { #1 }

```

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

```

1290 \stex_if_in_module:TF {
1291   % Nested module
1292   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1293   { ns } \l_stex_module_ns_str
1294   \str_set:Nx \l_stex_module_name_str {
1295     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1296     { name } / \l_stex_module_name_str
1297   }
1298   \str_if_empty:NT \l_stex_module_lang_str {
1299     \str_set:Nx \l_stex_module_lang_str {
1300       \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1301       { lang }
1302     }
1303   }
1304 }{
1305   % not nested:
1306   \str_if_empty:NT \l_stex_module_ns_str {
1307     \stex_modules_current_namespace:
1308     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1309       / { \l_stex_module_ns_str }
1310     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1311     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1312       \str_set:Nx \l_stex_module_ns_str {
1313         \stex_path_to_string:N \l_tmpa_seq
1314       }
1315     }
1316   }
1317 }

```

Next, we determine the language of the module:

```

1318 \str_if_empty:NT \l_stex_module_lang_str {
1319   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1320   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1321   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1322   \exp_args:No \str_if_eq:nnF \l_tmpa_str {tex} {
1323     \exp_args:No \str_if_eq:nnF \l_tmpa_str {dtx} {
1324       \exp_args:NNNo \seq_put_right:Nn \l_tmpa_seq \l_tmpa_str
1325     }
1326   }
1327   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1328   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be [<something>.]language
1329     \seq_pop_right:NN \l_tmpa_seq \l_stex_module_lang_str
1330     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1331       inferred~from~file~name}
1332   }
1333 }
1334
1335 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {

```

```

1336 \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1337 \l_tmpa_str {
1338   \ltx@ifpackageloaded{babel}{
1339     \exp_args:Nx \selectlanguage { \l_tmpa_str }
1340   }{}
1341 } {
1342   \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1343 }
1344 }}

```

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

```

1345 \str_if_empty:NTF \l_stex_module_sig_str {
1346   \exp_args:Nnx \prop_gset_from_keyval:cn {
1347     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1348   } {
1349     name      = \l_stex_module_name_str ,
1350     ns        = \l_stex_module_ns_str ,
1351     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1352     lang      = \l_stex_module_lang_str ,
1353     sig       = \l_stex_module_sig_str ,
1354     deprecate = \l_stex_module_deprecate_str ,
1355     meta      = \l_stex_module_meta_str
1356   }
1357   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1358   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1359   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1360   \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1361   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1362 \str_if_empty:NT \l_stex_module_meta_str {
1363   \str_set:Nx \l_stex_module_meta_str {
1364     \c_stex_metatheory_ns_str ? Metatheory
1365   }
1366 }
1367 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1368   \bool_set_true:N \l_stex_in_meta_bool
1369   \exp_args:Nx \stex_add_to_current_module:n {
1370     \bool_set_true:N \l_stex_in_meta_bool
1371     \stex_activate_module:n {\l_stex_module_meta_str}
1372     \bool_set_false:N \l_stex_in_meta_bool
1373   }
1374   \stex_activate_module:n {\l_stex_module_meta_str}
1375   \bool_set_false:N \l_stex_in_meta_bool
1376 }
1377 }{
1378   \str_if_empty:NT \l_stex_module_lang_str {
1379     \msg_error:nnxx{stex}{error/siglanguage}{
1380       \l_stex_module_ns_str?\l_stex_module_name_str
1381     }\l_stex_module_sig_str}
1382   }
1383   \stex_debug:nn{modules}{Signature~\l_stex_module_sig_str~for~\l_stex_module_ns_str?\l_st
1384   \stex_if_module_exists:nTF{\l_stex_module_ns_str?\l_stex_module_name_str}{

```

```

1385     \stex_debug:nn{modules}{(already exists)}
1386   }{
1387     \stex_debug:nn{modules}{(needs loading)}
1388     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1389     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1390     \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1391     \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1392     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1393     \str_set:Nx \l_tmpa_str {
1394       \stex_path_to_string:N \l_tmpa_seq /
1395       \l_tmpa_str . \l_stex_module_sig_str .tex
1396     }
1397     \IfFileExists \l_tmpa_str {
1398       \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1399         \str_clear:N \l_stex_current_module_str
1400         \seq_clear:N \l_stex_all_modules_seq
1401         \stex_debug:nn{modules}{Loading~signature}
1402       }
1403     }{
1404       \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1405     }
1406   }
1407   \stex_if_smsmode:F {
1408     \stex_activate_module:n {
1409       \l_stex_module_ns_str ? \l_stex_module_name_str
1410     }
1411   }
1412   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1413 }
1414 \str_if_empty:NF \l_stex_module_deprecate_str {
1415   \msg_warning:nnxx{stex}{warning/deprecated}{
1416     Module~\l_stex_current_module_str
1417   }{
1418     \l_stex_module_deprecate_str
1419   }
1420 }
1421 \seq_put_right:Nx \l_stex_all_modules_seq {
1422   \l_stex_module_ns_str ? \l_stex_module_name_str
1423 }
1424 \tl_clear:c{l__stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1425 }

```

(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}`

```

1426 \cs_new_protected:Nn \_stex_modules_begin_module: {
1427   \stex_reactivate_macro:N \STEXexport
1428   \stex_reactivate_macro:N \importmodule
1429   \stex_reactivate_macro:N \symdecl
1430   \stex_reactivate_macro:N \notation
1431   \stex_reactivate_macro:N \symdef
1432 }

```

```

1433 \stex_debug:nn{modules}{
1434   New~module:\
1435   Namespace:~\l_stex_module_ns_str\
1436   Name:~\l_stex_module_name_str\
1437   Language:~\l_stex_module_lang_str\
1438   Signature:~\l_stex_module_sig_str\
1439   Metatheory:~\l_stex_module_meta_str\
1440   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1441 }
1442
1443 \stex_if_do_html:T{
1444   \begin{stex_annotate_env} {theory} {
1445     \l_stex_module_ns_str ? \l_stex_module_name_str
1446   }
1447
1448   \stex_annotate_invisible:nnn{header}{} {
1449     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1450     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1451     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1452       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1453     }
1454     \str_if_empty:NF \smodulotype {
1455       \stex_annotate:nnn{type}{\smodulotype}{}
1456     }
1457   }
1458 }
1459 % TODO: Inherit metatheory for nested modules?
1460 }
1461 \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}

```

1462 \cs_new_protected:Nn \_stex_modules_end_module: {
1463   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}
1464   \_stex_reset_up_to_module:n \l_stex_current_module_str
1465   \stex_if_smsmode:T {
1466     \stex_persist:x {
1467       \prop_set_from_keyval:cn{c_stex_module\_l_stex_current_module_str _prop}{
1468         \exp_after:wN \prop_to_keyval:N \csname c_stex_module\_l_stex_current_module_str _pr
1469       }
1470       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _constants}{
1471         \seq_use:cn{c_stex_module\_l_stex_current_module_str _constants},
1472       }
1473       \seq_set_from_clist:cn{c_stex_module\_l_stex_current_module_str _imports}{
1474         \seq_use:cn{c_stex_module\_l_stex_current_module_str _imports},
1475       }
1476       \tl_set:cn {c_stex_module\_l_stex_current_module_str _code}
1477     }
1478     \exp_after:wN \let \exp_after:wN \l_tmpa_tl \csname c_stex_module\_l_stex_current_module
1479     \exp_after:wN \stex_persist:n \exp_after:wN { \exp_after:wN { \l_tmpa_tl } }
1480   }
1481 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1482 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1483 \NewDocumentEnvironment { smodule } { 0{} m } {
1484   \stex_module_setup:nn{#1}{#2}
1485   \par
1486   \stex_if_smsmode:F{
1487     \tl_if_empty:NF \smodulename {
1488       \exp_args:No \stex_document_title:n \smodulename
1489     }
1490     \tl_clear:N \l_tmpa_tl
1491     \clist_map_inline:Nn \smodulename {
1492       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1493         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1494       }
1495     }
1496     \tl_if_empty:NTF \l_tmpa_tl {
1497       \__stex_modules_smodule_start:
1498     }{
1499       \l_tmpa_tl
1500     }
1501   }
1502   \__stex_modules_begin_module:
1503   \str_if_empty:NF \smoduleid {
1504     \stex_ref_new_doc_target:n \smoduleid
1505   }
1506   \stex_smsmode_do:
1507 } {
1508   \__stex_modules_end_module:
1509   \stex_if_smsmode:F {
1510     \end{stex_annotate_env}
1511     \clist_set:No \l_tmpa_clist \smodulename
1512     \tl_clear:N \l_tmpa_tl
1513     \clist_map_inline:Nn \l_tmpa_clist {
1514       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1515         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1516       }
1517     }
1518     \tl_if_empty:NTF \l_tmpa_tl {
1519       \__stex_modules_smodule_end:
1520     }{
1521       \l_tmpa_tl
1522     }
1523   }
1524 }

```

\stexpatchmodule

```

1525 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1526 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1527
1528 \newcommand\stexpatchmodule[3] [] {
1529   \str_set:Nx \l_tmpa_str{ #1 }
1530   \str_if_empty:NTF \l_tmpa_str {

```

```

1531     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1532     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1533   }{
1534     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1535     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1536   }
1537 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1538 \NewDocumentCommand \STEXModule { m } {
1539   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1540   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1541   \tl_set:Nn \l_tmpa_tl {
1542     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1543   }
1544   \seq_map_inline:Nn \l_stex_all_modules_seq {
1545     \str_set:Nn \l_tmpb_str { ##1 }
1546     \str_if_eq:eeT { \l_tmpa_str } {
1547       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1548     } {
1549       \seq_map_break:n {
1550         \tl_set:Nn \l_tmpa_tl {
1551           \stex_invoke_module:n { ##1 }
1552         }
1553       }
1554     }
1555   }
1556   \l_tmpa_tl
1557 }

1558
1559 \cs_new_protected:Nn \stex_invoke_module:n {
1560   \stex_debug:nn{modules}{Invoking~module~#1}
1561   \peek_charcode_remove:NTF ! {
1562     \__stex_modules_invoke_uri:nN { #1 }
1563   } {
1564     \peek_charcode_remove:NTF ? {
1565       \__stex_modules_invoke_symbol:nn { #1 }
1566     } {
1567       \msg_error:nnx{stex}{error/syntax}{
1568         ?~or~!~expected~after~
1569         \c_backslash_str STEXModule{#1}
1570       }
1571     }
1572   }
1573 }

1574
1575 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1576   \str_set:Nn #2 { #1 }
1577 }

```



```

1578
1579 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1580   \stex_invoke_symbol:n{#1?#2}
1581 }

```

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

\stex_activate_module:n

```

1582 \bool_new:N \l_stex_in_meta_bool
1583 \bool_set_false:N \l_stex_in_meta_bool
1584 \cs_new_protected:Nn \stex_activate_module:n {
1585   \stex_debug:nn{modules}{Activating~module~#1}
1586   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1587     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1588     \use:c{ c_stex_module_#1_code }
1589   }
1590 }

```

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

```

1591 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1592 <*package>
1593
1594 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1595
```

28.1 SMS Mode

```
1596 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1597 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1598 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1599 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1600
1601 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1602   \makeatletter
1603   \makeatother
1604   \ExplSyntaxOn
1605   \ExplSyntaxOff
1606   \rustexBREAK
1607 }
1608
1609 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1610   \symdef
1611   \importmodule
1612   \notation
1613   \symdecl
1614   \STEXexport
1615   \inlineass
1616   \inlinedef
1617   \inlineex
1618   \endinput
1619   \setnotation
```

```

1620 \copynotation
1621 \assign
1622 \renamedekl
1623 \donotcopy
1624 \instantiate
1625 }
1626
1627 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1628   \tl_to_str:n {
1629     smodule,
1630     copymodule,
1631     interpretmodule,
1632     sdefinition,
1633     sexample,
1634     sassertion,
1635     sparagraph,
1636     mathstructure
1637   }
1638 }

```

(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`

```

1639 \bool_new:N \g__stex_smsmode_bool
1640 \bool_set_false:N \g__stex_smsmode_bool
1641 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1642   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1643 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1644 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn { \stex_suppress_html:n {
1645   \vbox_set:Nn \l_tmpa_box {
1646     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1647     \bool_gset_true:N \g__stex_smsmode_bool
1648     #2
1649     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1650   }
1651   \box_clear:N \l_tmpa_box
1652 } }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1653 \quark_new:N \q__stex_smsmode_break
1654
1655 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1656   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2}}
1657   \stex_smsmode_do:
1658 }
1659
1660 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1661   \_stex_modules_args:n{#1}

```

```

1662 \stex_if_in_module:F {
1663   \str_if_empty:NF \l_stex_module_sig_str {
1664     \stex_modules_current_namespace:
1665     \str_set:Nx \l_stex_module_name_str { #2 }
1666     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1667       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1668       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1669       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1670       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1671       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1672       \str_set:Nx \l_tmpa_str {
1673         \stex_path_to_string:N \l_tmpa_seq /
1674         \l_tmpa_str . \l_stex_module_sig_str .tex
1675       }
1676       \IfFileExists \l_tmpa_str {
1677         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1678       }{
1679         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1680       }
1681     }
1682   }
1683 }
1684 }
1685
1686 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1687   \stex_filestack_push:n{#1}
1688   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1689   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1690   % ----- new -----
1691   \__stex_smsmode_in_smsmode:nn{#1}{
1692     \let\importmodule\__stex_smsmode_importmodule:
1693     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1694     \let\__stex_modules_begin_module:\relax
1695     \let\__stex_modules_end_module:\relax
1696     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1697     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{module}}
1698     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1699     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1700     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1701     \everyeof{\q__stex_smsmode_break\noexpand}
1702     \expandafter\expandafter\expandafter
1703     \stex_smsmode_do:
1704     \csname @ @ input\endcsname "#1"\relax
1705
1706     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1707       \stex_filestack_push:n{##1}
1708       \expandafter\expandafter\expandafter
1709       \stex_smsmode_do:
1710       \csname @ @ input\endcsname "##1"\relax
1711       \stex_filestack_pop:
1712     }
1713   }
1714   % ----- new -----
1715   \__stex_smsmode_in_smsmode:nn{#1} {

```

```

1716 #2
1717 % ----- new -----
1718 \begingroup
1719 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1720 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1721   \stex_import_module_uri:nn ##1
1722   \stex_import_require_module:nnnn
1723   \l_stex_import_ns_str
1724   \l_stex_import_archive_str
1725   \l_stex_import_path_str
1726   \l_stex_import_name_str
1727 }
1728 \endgroup
1729 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1730 % ----- new -----
1731 \everyeof{\q__stex_smsmode_break\noexpand}
1732 \expandafter\expandafter\expandafter
1733 \stex_smsmode_do:
1734 \csname @ @ input\endcsname "#1"\relax
1735 }
1736 \stex_filestack_pop:
1737 }

```

(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:

```

1738 \cs_new_protected:Npn \stex_smsmode_do: {
1739   \stex_if_smsmode:T {
1740     \__stex_smsmode_do:w
1741   }
1742 }
1743 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1744   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1745     \expandafter\if\expandafter\relax\noexpand#1
1746     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1747     \else\expandafter\__stex_smsmode_do:w\fi
1748   }{
1749     \__stex_smsmode_do:w %#1
1750   }
1751 }
1752 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1753   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1754     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1755       #1\__stex_smsmode_do:w
1756     }{
1757       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1758         #1
1759       }{
1760         \cs_if_eq:NNTF \begin #1 {
1761           \__stex_smsmode_check_begin:n
1762         }{
1763           \cs_if_eq:NNTF \end #1 {
1764             \__stex_smsmode_check_end:n

```

```

1765         }{
1766         \__stex_smsmode_do:w
1767         }
1768     }
1769 }
1770 }
1771 }
1772 }
1773
1774 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1775 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1776 \begin{#1}
1777 }{
1778 \__stex_smsmode_do:w
1779 }
1780 }
1781 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1782 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1783 \end{#1}\__stex_smsmode_do:w
1784 }{
1785 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1786 }
1787 }

```

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

28.2 Inheritance

```

1788 <@@=stex_importmodule>

\stex_import_module_uri:nn

1789 \cs_new_protected:Nn \stex_import_module_uri:nn {
1790 \str_set:Nx \l_stex_import_archive_str { #1 }
1791 \str_set:Nn \l_stex_import_path_str { #2 }
1792
1793 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1794 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1795 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1796
1797 \stex_modules_current_namespace:
1798 \bool_lazy_all:nTF {
1799 {\str_if_empty_p:N \l_stex_import_archive_str}
1800 {\str_if_empty_p:N \l_stex_import_path_str}
1801 {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1802 }{
1803 \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1804 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1805 }{
1806 \str_if_empty:NT \l_stex_import_archive_str {
1807 \prop_if_exist:NT \l_stex_current_repository_prop {
1808 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1809 }
1810 }
1811 \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1812     \str_if_empty:NF \l_stex_import_path_str {
1813         \str_set:Nx \l_stex_import_ns_str {
1814             \l_stex_module_ns_str / \l_stex_import_path_str
1815         }
1816     }
1817 }{
1818     \stex_require_repository:n \l_stex_import_archive_str
1819     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1820     \l_stex_import_ns_str
1821     \str_if_empty:NF \l_stex_import_path_str {
1822         \str_set:Nx \l_stex_import_ns_str {
1823             \l_stex_import_ns_str / \l_stex_import_path_str
1824         }
1825     }
1826 }
1827 }
1828 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str
1829 \str_new:N \l_stex_import_name_str
1830 \str_new:N \l_stex_import_archive_str
1831 \str_new:N \l_stex_import_path_str
1832 \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 {\<ns>} {\<archive-ID>} {\<path>} {\<name>}
1833 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1834     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1835
1836         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1837
1838         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1839         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1840
1841         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1842
1843         % archive
1844         \str_set:Nx \l_tmpa_str { #2 }
1845         \str_if_empty:NTF \l_tmpa_str {
1846             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1847         } {
1848             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1849             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1850             \seq_put_right:Nn \l_tmpa_seq { source }
1851         }
1852
1853         % path
1854         \str_set:Nx \l_tmpb_str { #3 }
1855         \str_if_empty:NTF \l_tmpb_str {
1856             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1857

```

```

1858 \ltx@ifpackageloaded{babel} {
1859   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1860     { \language } \l_tmpb_str {
1861       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1862     }
1863   } {
1864     \str_clear:N \l_tmpb_str
1865   }
1866
1867   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1868   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1869     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1870   }{
1871     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1872     \IfFileExists{ \l_tmpa_str.tex }{
1873       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1874     }{
1875       % try english as default
1876       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1877       \IfFileExists{ \l_tmpa_str.en.tex }{
1878         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1879       }{
1880         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1881       }
1882     }
1883   }
1884
1885 } {
1886   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1887   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1888
1889   \ltx@ifpackageloaded{babel} {
1890     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1891       { \language } \l_tmpb_str {
1892         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1893       }
1894   } {
1895     \str_clear:N \l_tmpb_str
1896   }
1897
1898   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1899
1900   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1901   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1902     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1903   }{
1904     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1905     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1906       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1907     }{
1908       % try english as default
1909       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1910       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1911         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```



```

1912     }{
1913         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1914         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1915             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1916         }{
1917             %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1918             \IfFileExists{ \l_tmpa_str.tex }{
1919                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1920             }{
1921                 % try english as default
1922                 %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1923                 \IfFileExists{ \l_tmpa_str.en.tex }{
1924                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1925                 }{
1926                     \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1927                 }
1928             }
1929         }
1930     }
1931 }
1932 }
1933 }
1934
1935 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1936     \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1937         \seq_clear:N \l_stex_all_modules_seq
1938         \str_clear:N \l_stex_current_module_str
1939         \str_set:Nx \l_tmpb_str { #2 }
1940         \str_if_empty:NF \l_tmpb_str {
1941             \stex_set_current_repository:n { #2 }
1942         }
1943         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1944     }
1945
1946     \stex_if_module_exists:nF { #1 ? #4 } {
1947         \msg_error:nnx{stex}{error/unknownmodule}{
1948             #1?#4~(in~file~\g__stex_importmodule_file_str)
1949         }
1950     }
1951 }
1952
1953 }
1954 \stex_activate_module:n { #1 ? #4 }
1955 }

```

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

`\importmodule`

```

1956 \NewDocumentCommand \importmodule { 0{} m } {
1957     \stex_import_module_uri:nn { #1 } { #2 }
1958     \stex_debug:nn{modules}{Importing~module:~
1959         \l_stex_import_ns_str ? \l_stex_import_name_str
1960     }
1961     \stex_import_require_module:nnnn

```

```

1962 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1963 { \l_stex_import_path_str } { \l_stex_import_name_str }
1964 \stex_if_smsmode:F {
1965   \stex_annotate_invisible:nnn
1966   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1967 }
1968 \exp_args:Nx \stex_add_to_current_module:n {
1969   \stex_import_require_module:nnnn
1970   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1971   { \l_stex_import_path_str } { \l_stex_import_name_str }
1972 }
1973 \exp_args:Nx \stex_add_import_to_current_module:n {
1974   \l_stex_import_ns_str ? \l_stex_import_name_str
1975 }
1976 \stex_smsmode_do:
1977 \ignorespacesandpars
1978 }
1979 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1980 \NewDocumentCommand \usemodule { 0{} m } {
1981   \stex_if_smsmode:F {
1982     \stex_import_module_uri:nn { #1 } { #2 }
1983     \stex_import_require_module:nnnn
1984     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1985     { \l_stex_import_path_str } { \l_stex_import_name_str }
1986     \stex_annotate_invisible:nnn
1987     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1988   }
1989   \stex_smsmode_do:
1990   \ignorespacesandpars
1991 }

```

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

```

1992 \cs_new_protected:Nn \stex_csl_to_imports:Nn {
1993   \tl_if_empty:nF{#2}{
1994     \clist_set:Nn \l_tmpa_clist {#2}
1995     \clist_map_inline:Nn \l_tmpa_clist {
1996       \tl_if_head_eq_charcode:nNTF {##1}[{
1997         #1 ##1
1998       }{
1999         #1{##1}
2000       }
2001     }
2002   }
2003 }
2004 \cs_generate_variant:Nn \stex_csl_to_imports:Nn {No}
2005
2006
2007 </package>

```

Chapter 29

STEX -Symbols Implementation

```
2008 <*package>
2009
2010 %%%%%%%%%% symbols.dtx %%%%%%%%%%
2011
    Warnings and error messages
2012 \msg_new:nnn{stex}{error/wrongargs}{
2013   args~value~in~symbol~declaration~for~#1~
2014   needs~to~be~i,~a,~b~or~B,~but~#2~given
2015 }
2016 \msg_new:nnn{stex}{error/unknownsymbol}{
2017   No~symbol~#1~found!
2018 }
2019 \msg_new:nnn{stex}{error/seqlength}{
2020   Expected~#1~arguments;~got~#2!
2021 }
2022 \msg_new:nnn{stex}{error/unknownnotation}{
2023   Unknown~notation~#1~for~#2!
2024 }
```

29.1 Symbol Declarations

```
2025 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
2026 \cs_new_protected:Nn \stex_all_symbols:n {
2027   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
2028   \seq_map_inline:Nn \l_stex_all_modules_seq {
2029     \seq_map_inline:cn{c_stex_module_##1_constants}{
2030       \__stex_symdecl_all_symbols_cs{##1?####1}
2031     }
2032   }
2033 }
```

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

\STEXsymbol

```
2034 \NewDocumentCommand \STEXsymbol { m } {
2035   \stex_get_symbol:n { #1 }
2036   \exp_args:No
2037   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2038 }
```

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

symdecl arguments:

```
2039 \keys_define:nn { stex / symdecl } {
2040   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2041   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2042   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2043   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2044   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2045   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2046   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2047   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2048   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2049   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2050   assoc     .choices:nn =
2051             {bin,binl,binr,pre,conj,pwconj}
2052             {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2053 }
2054
2055 \bool_new:N \l_stex_symdecl_make_macro_bool
2056
2057 \cs_new_protected:Nn \__stex_symdecl_args:n {
2058   \str_clear:N \l_stex_symdecl_name_str
2059   \str_clear:N \l_stex_symdecl_args_str
2060   \str_clear:N \l_stex_symdecl_deprecate_str
2061   \str_clear:N \l_stex_symdecl_reorder_str
2062   \str_clear:N \l_stex_symdecl_assoctype_str
2063   \bool_set_false:N \l_stex_symdecl_local_bool
2064   \tl_clear:N \l_stex_symdecl_type_tl
2065   \tl_clear:N \l_stex_symdecl_definiens_tl
2066
2067   \keys_set:nn { stex / symdecl } { #1 }
2068 }
```

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

```
2069
2070 \NewDocumentCommand \symdecl { s m O{} } {
2071   \__stex_symdecl_args:n { #3 }
2072   \IfBooleanTF #1 {
2073     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2074   } {
2075     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2076   }
2077   \stex_symdecl_do:n { #2 }
2078   \stex_smsmode_do:
2079 }
```

```

2080
2081 \cs_new_protected:Nn \stex_symdecl_do:nn {
2082   \__stex_symdecl_args:n{#1}
2083   \bool_set_false:N \l_stex_symdecl_make_macro_bool
2084   \stex_symdecl_do:n{#2}
2085 }
2086
2087 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

2088 \cs_new_protected:Nn \stex_symdecl_do:n {
2089   \stex_if_in_module:F {
2090     % TODO throw error? some default namespace?
2091   }
2092
2093   \str_if_empty:NT \l_stex_symdecl_name_str {
2094     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2095   }
2096
2097   \prop_if_exist:cT { l_stex_symdecl_
2098     \l_stex_current_module_str ?
2099     \l_stex_symdecl_name_str
2100   _prop
2101 }{
2102   % TODO throw error (beware of circular dependencies)
2103 }
2104
2105 \prop_clear:N \l_tmpa_prop
2106 \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2107 \seq_clear:N \l_tmpa_seq
2108 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2109 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2110
2111 \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2112   \str_if_empty:NF \l_stex_module_deprecate_str {
2113     \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2114   }
2115 }
2116 \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2117
2118 \exp_args:No \stex_add_constant_to_current_module:n {
2119   \l_stex_symdecl_name_str
2120 }
2121
2122 % arity/args
2123 \int_zero:N \l_tmpb_int
2124
2125 \bool_set_true:N \l_tmpa_bool
2126 \str_map_inline:Nn \l_stex_symdecl_args_str {
2127   \token_case_meaning:NnF ##1 {
2128     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2129     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }

```

```

2130     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2131     {\tl_to_str:n a} {
2132         \bool_set_false:N \l_tmpa_bool
2133         \int_incr:N \l_tmpb_int
2134     }
2135     {\tl_to_str:n B} {
2136         \bool_set_false:N \l_tmpa_bool
2137         \int_incr:N \l_tmpb_int
2138     }
2139 }{
2140     \msg_error:nnxx{stex}{error/wrongargs}{
2141         \l_stex_current_module_str ?
2142         \l_stex_symdecl_name_str
2143     }{##1}
2144 }
2145 }
2146 \bool_if:NTF \l_tmpa_bool {
2147     % possibly numeric
2148     \str_if_empty:NTF \l_stex_symdecl_args_str {
2149         \prop_put:Nnn \l_tmpa_prop { args } {}
2150         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2151     }{
2152         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2153         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2154         \str_clear:N \l_tmpa_str
2155         \int_step_inline:nn \l_tmpa_int {
2156             \str_put_right:Nn \l_tmpa_str i
2157         }
2158         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2159     }
2160 } {
2161     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2162     \prop_put:Nnx \l_tmpa_prop { arity }
2163     { \str_count:N \l_stex_symdecl_args_str }
2164 }
2165 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2166
2167 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2168     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2169 }{
2170     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2171 }
2172
2173 % semantic macro
2174
2175 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2176     \exp_args:Nx \stex_do_up_to_module:n {
2177         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2178             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2179         }}
2180     }
2181 }
2182
2183 \stex_debug:nn{symbols}{New~symbol:~

```

```

2184 \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2185 Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2186 Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2187 Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2188 }
2189
2190 % circular dependencies require this:
2191 \stex_if_do_html:T {
2192   \stex_annotate_invisible:nnn {symdecl} {
2193     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2194   } {
2195     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2196       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2197     }
2198     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$}
2199     \prop_item:Nn \l_tmpa_prop { args }
2200   }
2201   \stex_annotate_invisible:nnn{macroname}{#1}{}
2202   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2203     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2204   }
2205   \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2206     \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2207   }
2208   \str_if_empty:NF \l_stex_symdecl_reorder_str {
2209     \stex_annotate_invisible:nnn{reorder_args}{\l_stex_symdecl_reorder_str}{}
2210   }
2211 }
2212 }
2213 }
2214 \prop_if_exist:cF {
2215   \l_stex_symdecl_
2216   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2217   _prop
2218 } {
2219   \bool_if:NTF \l_stex_symdecl_local_bool \stex_do_up_to_module:x \stex_execute_in_module:
2220     \__stex_symdecl_restore_symbol:nnnnnnn
2221       {\l_stex_symdecl_name_str}
2222       { \prop_item:Nn \l_tmpa_prop {args} }
2223       { \prop_item:Nn \l_tmpa_prop {arity} }
2224       { \prop_item:Nn \l_tmpa_prop {assoc} }
2225       { \prop_item:Nn \l_tmpa_prop {defined} }
2226       {\bool_if:NT \l_stex_symdecl_make_macro_bool {#1} }
2227       {\l_stex_current_module_str}
2228   }
2229 }
2230 }
2231 \cs_new_protected:Nn \__stex_symdecl_restore_symbol:nnnnnnn {
2232   \prop_clear:N \l_tmpa_prop
2233   \prop_put:Nnn \l_tmpa_prop { module } { #7 }
2234   \prop_put:Nnn \l_tmpa_prop { name } { #1 }
2235   \prop_put:Nnn \l_tmpa_prop { args } { #2 }
2236   \prop_put:Nnn \l_tmpa_prop { arity } { #3 }
2237   \prop_put:Nnn \l_tmpa_prop { assoc } { #4 }

```

```

2238 \prop_put:Nnn \l_tmpa_prop { defined } { #5 }
2239 \tl_if_empty:nF{#6}{
2240   \tl_set:cx{#6}{\stex_invoke_symbol:n{\detokenize{#7 ? #1}}}
2241 }
2242 \prop_set_eq:cN{l_stex_symdecl_ \detokenize{#7 ? #1} _prop}\l_tmpa_prop
2243 \seq_clear:c{l_stex_symdecl_ \detokenize{#7 ? #1} _notations}
2244 }

```

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

`\stex_get_symbol:n`

```

2245 \str_new:N \l_stex_get_symbol_uri_str
2246
2247 \cs_new_protected:Nn \stex_get_symbol:n {
2248   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2249     \tl_set:Nn \l_tmpa_tl { #1 }
2250     \__stex_symdecl_get_symbol_from_cs:
2251   }{
2252     % argument is a string
2253     % is it a command name?
2254     \cs_if_exist:cTF { #1 }{
2255       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2256       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2257       \str_if_empty:NTF \l_tmpa_str {
2258         \exp_args:Nx \cs_if_eq:NNTF {
2259           \tl_head:N \l_tmpa_tl
2260         } \stex_invoke_symbol:n {
2261           \__stex_symdecl_get_symbol_from_cs:
2262         }{
2263           \__stex_symdecl_get_symbol_from_string:n { #1 }
2264         }
2265       } {
2266         \__stex_symdecl_get_symbol_from_string:n { #1 }
2267       }
2268     }{
2269       % argument is not a command name
2270       \__stex_symdecl_get_symbol_from_string:n { #1 }
2271       % \l_stex_all_symbols_seq
2272     }
2273   }
2274   \str_if_eq:eeF {
2275     \prop_item:cn {
2276       l_stex_symdecl_\l_stex_get_symbol_uri_str _prop
2277     }{ deprecate }
2278   }{
2279     \msg_warning:nxxx{stex}{warning/deprecated}{
2280       Symbol~\l_stex_get_symbol_uri_str
2281     }{
2282       \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{ deprecate }
2283     }
2284   }
2285 }
2286
2287 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {

```



```

2288 \tl_set:Nn \l_tmpa_tl {
2289   \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2290 }
2291 \str_set:Nn \l_tmpa_str { #1 }
2292
2293 %\int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2294
2295 \str_if_in:NnTF \l_tmpa_str ? {
2296   \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
2297   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
2298   \str_set:Nx \l_tmpb_str {\seq_use:Nn \l_tmpa_seq ?}
2299 }{
2300   \str_clear:N \l_tmpb_str
2301 }
2302 \str_if_empty:NNTF \l_tmpb_str {
2303   \seq_map_inline:Nn \l_stex_all_modules_seq {
2304     \seq_map_inline:cn{c_stex_module_###1_constants}{
2305       \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2306         \seq_map_break:n{\seq_map_break:n{
2307           \tl_set:Nn \l_tmpa_tl {
2308             \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2309           }
2310         }}
2311       }
2312     }
2313   }
2314 }{
2315   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpb_str }
2316   \seq_map_inline:Nn \l_stex_all_modules_seq {
2317     \str_if_eq:eeT{ \l_tmpb_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2318       \seq_map_inline:cn{c_stex_module_###1_constants}{
2319         \exp_args:Nno \str_if_eq:nnT{####1} \l_tmpa_str {
2320           \seq_map_break:n{\seq_map_break:n{
2321             \tl_set:Nn \l_tmpa_tl {
2322               \str_set:Nn \l_stex_get_symbol_uri_str { ##1 ? ####1 }
2323             }
2324           }}
2325         }
2326       }
2327     }
2328   }
2329 }
2330
2331 \l_tmpa_tl
2332 }
2333
2334 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2335   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2336     { \tl_tail:N \l_tmpa_tl }
2337   \tl_if_single:NNTF \l_tmpa_tl {
2338     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2339       \exp_after:wN \str_set:Nn \exp_after:wN
2340         \l_stex_get_symbol_uri_str \l_tmpa_tl
2341     }{

```

```

2342     % TODO
2343     % tail is not a single group
2344   }
2345 }{
2346   % TODO
2347   % tail is not a single group
2348 }
2349 }

```

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

29.2 Notations

```

2350 <@=stex_notation>

notation arguments:
2351 \keys_define:nn { stex / notation } {
2352   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2353   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2354   prec     .str_set_x:N = \l__stex_notation_prec_str ,
2355   op       .tl_set:N = \l__stex_notation_op_tl ,
2356   primary .bool_set:N = \l__stex_notation_primary_bool ,
2357   primary .default:n = {true} ,
2358   unknown .code:n = \str_set:Nx
2359             \l__stex_notation_variant_str \l_keys_key_str
2360 }
2361
2362 \cs_new_protected:Nn \stex_notation_args:n {
2363   % \str_clear:N \l__stex_notation_lang_str
2364   \str_clear:N \l__stex_notation_variant_str
2365   \str_clear:N \l__stex_notation_prec_str
2366   \tl_clear:N \l__stex_notation_op_tl
2367   \bool_set_false:N \l__stex_notation_primary_bool
2368
2369   \keys_set:nn { stex / notation } { #1 }
2370 }

\notation

2371 \NewDocumentCommand \notation { s m 0{}} {
2372   \stex_notation_args:n { #3 }
2373   \tl_clear:N \l_stex_symdecl_definiens_tl
2374   \stex_get_symbol:n { #2 }
2375   \tl_set:Nn \l_stex_notation_after_do_tl {
2376     \__stex_notation_final:
2377     \IfBooleanTF#1{
2378       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2379     }{}
2380     \stex_smsmode_do:\ignorespacesandpars
2381   }
2382   \stex_notation_do:nnnnn
2383   { \prop_item:cn {\l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2384   { \prop_item:cn { \l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2385   { \l__stex_notation_variant_str }
2386   { \l__stex_notation_prec_str }

```

```

2387 }
2388 \stex_deactivate_macro:Nn \notation {module-environments}

```

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

\stex_notation_do:nnnnn

```

2389 \seq_new:N \l__stex_notation_precedences_seq
2390 \tl_new:N \l__stex_notation_opprec_tl
2391 \int_new:N \l__stex_notation_currarg_int
2392 \tl_new:N \stex_symbol_after_invokation_tl
2393
2394 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2395   \let\l_stex_current_symbol_str\relax
2396   \seq_clear:N \l__stex_notation_precedences_seq
2397   \tl_clear:N \l__stex_notation_opprec_tl
2398   \str_set:Nx \l__stex_notation_args_str { #1 }
2399   \str_set:Nx \l__stex_notation_arity_str { #2 }
2400   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2401   \str_set:Nx \l__stex_notation_prec_str { #4 }
2402
2403   % precedences
2404   \str_if_empty:NTF \l__stex_notation_prec_str {
2405     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2406       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2407     }{
2408       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2409     }
2410   } {
2411     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2412       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2413       \int_step_inline:nn { \l__stex_notation_arity_str } {
2414         \exp_args:NNo
2415         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2416       }
2417     }{
2418       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2419       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2420         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2421         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2422           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2423             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2424           \seq_map_inline:Nn \l_tmpa_seq {
2425             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2426           }
2427         }
2428       }{
2429         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2430           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2431         }{
2432           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2433         }
2434       }
2435     }
2436   }

```

```

2437 \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2438 \int_step_inline:nn { \l__stex_notation_arity_str } {
2439   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2440     \exp_args:NNo
2441     \seq_put_right:No \l__stex_notation_precedences_seq {
2442       \l__stex_notation_opprec_tl
2443     }
2444   }
2445 }
2446 }
2447 \tl_clear:N \l_stex_notation_dummyargs_tl
2448
2449 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2450   \exp_args:NNe
2451   \cs_set:Npn \l_stex_notation_macrocode_cs {
2452     \stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2453     { \l__stex_notation_suffix_str }
2454     { \l__stex_notation_opprec_tl }
2455     { \exp_not:n { #5 } }
2456   }
2457   \l_stex_notation_after_do_tl
2458 }{
2459   \str_if_in:NnTF \l__stex_notation_args_str b {
2460     \exp_args:Nne \use:nn
2461     {
2462       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2463       \cs_set:Npn \l__stex_notation_arity_str } { {
2464         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2465         { \l__stex_notation_suffix_str }
2466         { \l__stex_notation_opprec_tl }
2467         { \exp_not:n { #5 } }
2468       } }
2469   }{
2470     \str_if_in:NnTF \l__stex_notation_args_str B {
2471       \exp_args:Nne \use:nn
2472       {
2473         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2474         \cs_set:Npn \l__stex_notation_arity_str } { {
2475           \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2476           { \l__stex_notation_suffix_str }
2477           { \l__stex_notation_opprec_tl }
2478           { \exp_not:n { #5 } }
2479         } }
2480     }{
2481       \exp_args:Nne \use:nn
2482       {
2483         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2484         \cs_set:Npn \l__stex_notation_arity_str } { {
2485           \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2486           { \l__stex_notation_suffix_str }
2487           { \l__stex_notation_opprec_tl }
2488           { \exp_not:n { #5 } }
2489         } }
2490     }

```

```

2491     }
2492
2493     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2494     \int_zero:N \l__stex_notation_currarg_int
2495     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2496     \__stex_notation_arguments:
2497   }
2498 }

```

(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

```

2499 \cs_new_protected:Nn \__stex_notation_arguments: {
2500   \int_incr:N \l__stex_notation_currarg_int
2501   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2502     \l_stex_notation_after_do_tl
2503   }{
2504     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2505     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2506     \str_if_eq:VnTF \l_tmpa_str a {
2507       \__stex_notation_argument_assoc:nn{a}
2508     }{
2509       \str_if_eq:VnTF \l_tmpa_str B {
2510         \__stex_notation_argument_assoc:nn{B}
2511       }{
2512         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2513         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2514           { \stex_term_math_arg:nnn
2515             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2516             { \l_tmpb_str }
2517             { ###\int_use:N \l__stex_notation_currarg_int }
2518           }
2519         }
2520         \__stex_notation_arguments:
2521       }
2522     }
2523   }
2524 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:nn

```

2525 \cs_new_protected:Nn \__stex_notation_argument_assoc:nn {
2526
2527   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2528     {\l__stex_notation_arity_str}{
2529     #2
2530   }
2531   \int_zero:N \l_tmpa_int
2532   \tl_clear:N \l_tmpa_tl
2533   \str_map_inline:Nn \l__stex_notation_args_str {
2534     \int_incr:N \l_tmpa_int
2535     \tl_put_right:Nx \l_tmpa_tl {
2536       \str_if_eq:nnTF {##1}{a}{ { } }{ }

```

```

2537     \str_if_eq:nnTF {##1}{B}{ } }{
2538         {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2539         }
2540     }
2541 }
2542 }
2543 \exp_after:wN\exp_after:wN\exp_after:wN \def
2544 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2545 \exp_after:wN\exp_after:wN\exp_after:wN ##
2546 \exp_after:wN\exp_after:wN\exp_after:wN 1
2547 \exp_after:wN\exp_after:wN\exp_after:wN ##
2548 \exp_after:wN\exp_after:wN\exp_after:wN 2
2549 \exp_after:wN\exp_after:wN\exp_after:wN {
2550     \exp_after:wN \exp_after:wN \exp_after:wN
2551     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2552     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2553 }
2554 }
2555
2556 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2557 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2558     \_stex_term_math_assoc_arg:nnnn
2559     { #1\int_use:N \l__stex_notation_currarg_int }
2560     { \l_tmpa_str }
2561     { #####\int_use:N \l__stex_notation_currarg_int }
2562     { \l_tmpa_cs {####1} {####2} }
2563 } }
2564 \__stex_notation_arguments:
2565 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2566 \cs_new_protected:Nn \_stex_notation_restore_notation:nnnnn {
2567     \cs_generate_from_arg_count:cNnn{stex_notation_\detokenize{#1} \c_hash_str \detokenize{#2}
2568     \cs_set_nopar:Npn {#3}{#4}
2569     \tl_if_empty:nF {#5}{
2570         \tl_set:cn{stex_op_notation_\detokenize{#1} \c_hash_str \detokenize{#2}_cs}{\comp{ #5 }
2571     }
2572     \seq_if_exist:cT { l_stex_symdecl_\detokenize{#1} _notations }{
2573         \seq_put_right:cx { l_stex_symdecl_\detokenize{#1} _notations } { \detokenize{#2} }
2574     }
2575 }
2576
2577 \cs_new_protected:Nn \_stex_notation_final: {
2578
2579     \stex_execute_in_module:x {
2580         \_stex_notation_restore_notation:nnnnn
2581         {\l_stex_get_symbol_uri_str}
2582         {\l_stex_notation_suffix_str}
2583         {\l__stex_notation_arity_str}
2584     {
2585         \exp_after:wN \exp_after:wN \exp_after:wN
2586         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN

```

```

2587     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2588   }
2589   {\exp_args:No \exp_not:n \l__stex_notation_op_tl }
2590 }
2591
2592 \stex_debug:nn{symbols}{
2593   Notation~\l__stex_notation_suffix_str
2594   ~for~\l_stex_get_symbol_uri_str^^J
2595   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2596   Argument~precedences:~
2597     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2598   Notation: \cs_meaning:c {
2599     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2600     \l__stex_notation_suffix_str
2601     _cs
2602   }
2603 }
2604 % HTML annotations
2605 \stex_if_do_html:T {
2606   \stex_annotate_invisible:nnn { notation }
2607   { \l_stex_get_symbol_uri_str } {
2608     \stex_annotate_invisible:nnn { notationfragment }
2609     { \l__stex_notation_suffix_str }{}
2610     \stex_annotate_invisible:nnn { precedence }
2611     { \l__stex_notation_prec_str }{}
2612
2613     \int_zero:N \l_tmpa_int
2614     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2615     \tl_clear:N \l_tmpa_tl
2616     \int_step_inline:nn { \l__stex_notation_arity_str }{
2617       \int_incr:N \l_tmpa_int
2618       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2619       \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_rema
2620       \str_if_eq:VnTF \l_tmpb_str a {
2621         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2622           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2623           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2624         } }
2625       }{
2626         \str_if_eq:VnTF \l_tmpb_str B {
2627           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2628             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2629             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2630           } }
2631         }{
2632           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2633             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2634           } }
2635       }
2636     }
2637   }
2638   \stex_annotate_invisible:nnn { notationcomp }{}{
2639     \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2640     $ \exp_args:Nno \use:nn { \use:c {

```

```

2641         stex_notation_ \l_stex_current_symbol_str
2642         \c_hash_str \l__stex_notation_suffix_str _cs
2643     } } { \l_tmpa_tl } $
2644 }
2645 }
2646 }
2647 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2648 \keys_define:nn { stex / setnotation } {
2649   % lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2650   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2651   unknown .code:n      = \str_set:Nx
2652       \l__stex_notation_variant_str \l_keys_key_str
2653 }
2654
2655 \cs_new_protected:Nn \stex_setnotation_args:n {
2656   % \str_clear:N \l__stex_notation_lang_str
2657   \str_clear:N \l__stex_notation_variant_str
2658   \keys_set:nn { stex / setnotation } { #1 }
2659 }
2660
2661 \cs_new_protected:Nn \__stex_notation_setnotation:nn {
2662   \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2663     \seq_remove_all:cn { l_stex_symdecl_#1_notations }{ #2 }
2664     \seq_put_left:cn { l_stex_symdecl_#1_notations }{ #2 }
2665   }
2666 }
2667
2668 \cs_new_protected:Nn \stex_setnotation:n {
2669   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1_notations }
2670   { \l__stex_notation_variant_str }{
2671     \stex_execute_in_module:x{ \__stex_notation_setnotation:nn {#1}{\l__stex_notation_vari
2672     \stex_debug:nn {notations}{
2673       Setting~default~notation~
2674       {\l__stex_notation_variant_str }~for~
2675       #1 \\
2676       \expandafter\meaning\csname
2677       l_stex_symdecl_#1_notations\endcsname
2678     }
2679   }{
2680     \msg_error:nnxx{stex}{unknownnotation}{\l__stex_notation_variant_str}{#1}
2681   }
2682 }
2683
2684 \NewDocumentCommand \setnotation {m m} {
2685   \stex_get_symbol:n { #1 }
2686   \stex_setnotation_args:n { #2 }
2687   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2688   \stex_smsmode_do:\ignorespacesandpars
2689 }
2690

```



```

2691 \cs_new_protected:Nn \stex_copy_notations:nn {
2692   \stex_debug:nn {notations}{
2693     Copying~notations~from~#2~to~#1\\
2694     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2695   }
2696   \tl_clear:N \l_tmpa_tl
2697   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2698     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2699   }
2700   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2701     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2702     \edef \l_tmpa_tl {
2703       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2704       \exp_after:wN\exp_after:wN\exp_after:wN {
2705         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2706       }
2707     }
2708
2709     \stex_execute_in_module:x {
2710       \__stex_notation_restore_notation:nnnnn
2711       {#1}{##1}
2712       { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }
2713       { \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl} }
2714       {
2715         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2716           \exp_args:NNo\exp_args:No\exp_not:n{\csname stex_op_notation_ #2\c_hash_str ##1
2717         }
2718       }
2719     }
2720   }
2721 }
2722
2723 \NewDocumentCommand \copynotation {m m} {
2724   \stex_get_symbol:n { #1 }
2725   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2726   \stex_get_symbol:n { #2 }
2727   \exp_args:Noo
2728   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2729   \stex_smsmode_do:\ignorespacesandpars
2730 }
2731

```

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

\symdef

```

2732 \keys_define:nn { stex / symdef } {
2733   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2734   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2735   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2736   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2737   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2738   reorder   .str_set_x:N = \l_stex_symdecl_reorder_str ,
2739   op        .tl_set:N = \l__stex_notation_op_tl ,
2740   % lang     .str_set_x:N = \l__stex_notation_lang_str ,

```

```

2741   variant .str_set_x:N = \l__stex_notation_variant_str ,
2742   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2743   assoc   .choices:nn =
2744     {bin,binl,binr,pre,conj,pwconj}
2745     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2746   unknown .code:n      = \str_set:Nx
2747     \l__stex_notation_variant_str \l_keys_key_str
2748 }
2749
2750 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2751   \str_clear:N \l_stex_symdecl_name_str
2752   \str_clear:N \l_stex_symdecl_args_str
2753   \str_clear:N \l_stex_symdecl_assoctype_str
2754   \str_clear:N \l_stex_symdecl_reorder_str
2755   \bool_set_false:N \l_stex_symdecl_local_bool
2756   \tl_clear:N \l_stex_symdecl_type_tl
2757   \tl_clear:N \l_stex_symdecl_definiens_tl
2758   % \str_clear:N \l__stex_notation_lang_str
2759   \str_clear:N \l__stex_notation_variant_str
2760   \str_clear:N \l__stex_notation_prec_str
2761   \tl_clear:N \l__stex_notation_op_tl
2762
2763   \keys_set:nn { stex / symdef } { #1 }
2764 }
2765
2766 \NewDocumentCommand \symdef { m O{} } {
2767   \__stex_notation_symdef_args:n { #2 }
2768   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2769   \stex_symdecl_do:n { #1 }
2770   \tl_set:Nn \l_stex_notation_after_do_tl {
2771     \__stex_notation_final:
2772     \stex_smsmode_do:\ignorespacesandpars
2773   }
2774   \str_set:Nx \l_stex_get_symbol_uri_str {
2775     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2776   }
2777   \exp_args:Nx \stex_notation_do:nnnnn
2778     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { args } }
2779     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop } { arity } }
2780     { \l__stex_notation_variant_str }
2781     { \l__stex_notation_prec_str }
2782 }
2783 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

29.3 Variables

```

2784 <@@=stex_variables>
2785
2786 \keys_define:nn { stex / vardef } {
2787   name .str_set_x:N = \l__stex_variables_name_str ,
2788   args .str_set_x:N = \l__stex_variables_args_str ,
2789   type .tl_set:N    = \l__stex_variables_type_tl ,

```

```

2790 def      .tl_set:N      = \l__stex_variables_def_tl ,
2791 op       .tl_set:N      = \l__stex_variables_op_tl ,
2792 prec     .str_set_x:N    = \l__stex_variables_prec_str ,
2793 assoc    .choices:nn    =
2794     {bin,binl,binr,pre,conj,pwconj}
2795     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2796 bind     .choices:nn    =
2797     {forall,exists}
2798     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2799 }
2800
2801 \cs_new_protected:Nn \__stex_variables_args:n {
2802   \str_clear:N \l__stex_variables_name_str
2803   \str_clear:N \l__stex_variables_args_str
2804   \str_clear:N \l__stex_variables_prec_str
2805   \str_clear:N \l__stex_variables_assoctype_str
2806   \str_clear:N \l__stex_variables_bind_str
2807   \tl_clear:N \l__stex_variables_type_tl
2808   \tl_clear:N \l__stex_variables_def_tl
2809   \tl_clear:N \l__stex_variables_op_tl
2810
2811   \keys_set:nn { stex / vardef } { #1 }
2812 }
2813
2814 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2815   \__stex_variables_args:n {#2}
2816   \str_if_empty:NT \l__stex_variables_name_str {
2817     \str_set:Nx \l__stex_variables_name_str { #1 }
2818   }
2819   \prop_clear:N \l_tmpa_prop
2820   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2821
2822   \int_zero:N \l_tmpb_int
2823   \bool_set_true:N \l_tmpa_bool
2824   \str_map_inline:Nn \l__stex_variables_args_str {
2825     \token_case_meaning:NnF ##1 {
2826       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2827       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2828       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2829       {\tl_to_str:n a} {
2830         \bool_set_false:N \l_tmpa_bool
2831         \int_incr:N \l_tmpb_int
2832       }
2833       {\tl_to_str:n B} {
2834         \bool_set_false:N \l_tmpa_bool
2835         \int_incr:N \l_tmpb_int
2836       }
2837     }{
2838       \msg_error:nnxx{stex}{error/wrongargs}{
2839         variable~\l__stex_variables_name_str
2840       }{##1}
2841     }
2842   }
2843   \bool_if:NTF \l_tmpa_bool {

```

```

2844 % possibly numeric
2845 \str_if_empty:NTF \l__stex_variables_args_str {
2846   \prop_put:Nnn \l_tmpa_prop { args } {}
2847   \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2848 }{
2849   \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2850   \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2851   \str_clear:N \l_tmpa_str
2852   \int_step_inline:nn \l_tmpa_int {
2853     \str_put_right:Nn \l_tmpa_str i
2854   }
2855   \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2856   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2857 }
2858 } {
2859   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2860   \prop_put:Nnx \l_tmpa_prop { arity }
2861     { \str_count:N \l__stex_variables_args_str }
2862 }
2863 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2864 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2865
2866 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2867
2868 \tl_if_empty:NF \l__stex_variables_op_tl {
2869   \cs_set:cpx {
2870     stex_var_op_notation_\l__stex_variables_name_str_cs
2871   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2872 }
2873
2874 \tl_set:Nn \l_stex_notation_after_do_tl {
2875   \exp_args:Nne \use:nn {
2876     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2877     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2878   } {{
2879     \exp_after:wN \exp_after:wN \exp_after:wN
2880     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2881     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2882   }}
2883 \stex_if_do_html:T {
2884   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2885     \stex_annotate_invisible:nnn { precedence }
2886       { \l__stex_variables_prec_str }{}
2887   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2888     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2889     \stex_annotate_invisible:nnn{macroname}{#1}{
2890   \tl_if_empty:NF \l__stex_variables_def_tl {
2891     \stex_annotate_invisible:nnn{definiens}{
2892       {\l__stex_variables_def_tl$}
2893   }
2894   \str_if_empty:NF \l__stex_variables_assoctype_str {
2895     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2896   }
2897   \str_if_empty:NF \l__stex_variables_bind_str {

```

```

2898     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{ }
2899   }
2900   \int_zero:N \l_tmpa_int
2901   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2902   \tl_clear:N \l_tmpa_tl
2903   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2904     \int_incr:N \l_tmpa_int
2905     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2906     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2907     \str_if_eq:VnTF \l_tmpb_str a {
2908       \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2909         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2910         \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2911       } }
2912     }{
2913       \str_if_eq:VnTF \l_tmpb_str B {
2914         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2915           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2916           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2917         } }
2918       }{
2919         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2920           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
2921         } }
2922       }
2923     }
2924   }
2925   \stex_annotate_invisible:nnn { notationcomp }{ }{
2926     \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2927     $ \exp_args:Nno \use:nn { \use:c {
2928       stex_var_notation_\l__stex_variables_name_str_cs
2929     } } { \l_tmpa_tl } $
2930   }
2931 }
2932 }\ignorespacesandpars
2933 }
2934
2935 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2936 }
2937
2938 \cs_new:Nn \_stex_reset:N {
2939   \tl_if_exist:NTF #1 {
2940     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2941   }{
2942     \let \exp_not:N #1 \exp_not:N \undefined
2943   }
2944 }
2945
2946 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2947   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2948   \exp_args:Nnx \use:nn {
2949     % TODO
2950     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2951       #2

```

```

2952     }
2953   }{
2954     \_stex_reset:N \varnot
2955     \_stex_reset:N \vartype
2956     \_stex_reset:N \vardefi
2957   }
2958 }
2959
2960 \NewDocumentCommand \vardef { s } {
2961   \IfBooleanTF#1 {
2962     \__stex_variables_do_complex:nn
2963   }{
2964     \__stex_variables_do_simple:nnn
2965   }
2966 }
2967
2968 \NewDocumentCommand \svar { 0{} m }{
2969   \tl_if_empty:nTF {#1}{
2970     \str_set:Nn \l_tmpa_str { #2 }
2971   }{
2972     \str_set:Nn \l_tmpa_str { #1 }
2973   }
2974   \_stex_term_omv:nn {
2975     var://\l_tmpa_str
2976   }{
2977     \exp_args:Nnx \use:nn {
2978       \def\comp{\_varcomp}
2979       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2980       \comp{ #2 }
2981     }{
2982       \_stex_reset:N \comp
2983       \_stex_reset:N \l_stex_current_symbol_str
2984     }
2985   }
2986 }
2987
2988
2989
2990 \keys_define:nn { stex / varseq } {
2991   name .str_set_x:N = \l__stex_variables_name_str ,
2992   args .int_set:N   = \l__stex_variables_args_int ,
2993   type .tl_set:N    = \l__stex_variables_type_tl ,
2994   mid .tl_set:N     = \l__stex_variables_mid_tl ,
2995   bind .choices:nn =
2996     {forall,exists}
2997     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2998 }
2999
3000 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3001   \str_clear:N \l__stex_variables_name_str
3002   \int_set:Nn \l__stex_variables_args_int 1
3003   \tl_clear:N \l__stex_variables_type_tl
3004   \str_clear:N \l__stex_variables_bind_str
3005 }

```

```

3006 \keys_set:nn { stex / varseq } { #1 }
3007 }
3008
3009 \NewDocumentCommand \varseq {m O{} m m m}{
3010 \__stex_variables_seq_args:n { #2 }
3011 \str_if_empty:NT \l__stex_variables_name_str {
3012 \str_set:Nx \l__stex_variables_name_str { #1 }
3013 }
3014 \prop_clear:N \l_tmpa_prop
3015 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3016
3017 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3018 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3019 \msg_error:nnxx{stex}{error/seqlength}
3020 {\int_use:N \l__stex_variables_args_int}
3021 {\seq_count:N \l_tmpa_seq}
3022 }
3023 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3024 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3025 \msg_error:nnxx{stex}{error/seqlength}
3026 {\int_use:N \l__stex_variables_args_int}
3027 {\seq_count:N \l_tmpb_seq}
3028 }
3029 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3030 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3031
3032 \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3033 \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3034
3035 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3036 \int_step_inline:nn \l__stex_variables_args_int {
3037 \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
3038 }
3039 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3040 \tl_put_right:Nn \l_tmpa_tl {\,ellipses,}
3041 \tl_if_empty:NF \l__stex_variables_mid_tl {
3042 \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3043 \tl_put_right:Nn \l_tmpa_tl {\,ellipses,}
3044 }
3045 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3046 \int_step_inline:nn \l__stex_variables_args_int {
3047 \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3048 }
3049 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3050 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3051
3052
3053 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3054
3055 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3056
3057 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3058
3059 \int_step_inline:nn \l__stex_variables_args_int {

```

```

3060 \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3061   \_stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3062 }}
3063 }
3064
3065 \tl_set:Nx \l_tmpa_tl {
3066   \_stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3067     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3068   }
3069 }
3070
3071 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3072
3073 \exp_args:Nno \use:nn {
3074   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3075   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3076
3077   \stex_debug:nn{sequences}{New~Sequence:~
3078     \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3079     \prop_to_keyval:N \l_tmpa_prop
3080   }
3081   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3082     \tl_if_empty:NF \l__stex_variables_type_tl {
3083       \stex_annotate:nnn {type}{\}{\${\seqtype\l__stex_variables_type_tl$}
3084     }
3085     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3086     \str_if_empty:NF \l__stex_variables_bind_str {
3087       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3088     }
3089   }}
3090
3091   \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3092   \ignorespacesandpars
3093 }
3094
3095 </package>

```


Chapter 30

STEX -Terms Implementation

```
3096 <*package>
3097
3098 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3099
3100 <@@=stex_terms>
3101
3102   Warnings and error messages
3103   \msg_new:nnn{stex}{error/nonotation}{
3104     Symbol~#1~invoked,~but~has~no~notation#2!
3105   }
3106   \msg_new:nnn{stex}{error/notationarg}{
3107     Error~in~parsing~notation~#1
3108   }
3109   \msg_new:nnn{stex}{error/noop}{
3110     Symbol~#1~has~no~operator~notation~for~notation~#2
3111   }
3112   \msg_new:nnn{stex}{error/notallowed}{
3113     Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3114   }
3115   \msg_new:nnn{stex}{error/doubleargument}{
3116     Argument~#1~of~symbol~#2~already~assigned
3117   }
3118   \msg_new:nnn{stex}{error/overarity}{
3119     Argument~#1~invalid~for~symbol~#2~with~arity~#3
3120   }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3120
3121
3122 \bool_new:N \l_stex_allow_semantic_bool
3123 \bool_set_true:N \l_stex_allow_semantic_bool
3124
```

```

3125 \cs_new_protected:Nn \stex_invoke_symbol:n {
3126   \bool_if:NTF \l_stex_allow_semantic_bool {
3127     \str_if_eq:eeF {
3128       \prop_item:cn {
3129         l_stex_symdecl_#1_prop
3130       }{ deprecate }
3131     }{}{
3132       \msg_warning:nxxx{stex}{warning/deprecated}{
3133         Symbol~#1
3134       }{
3135         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3136       }
3137     }
3138     \if_mode_math:
3139       \exp_after:wN \__stex_terms_invoke_math:n
3140     \else:
3141       \exp_after:wN \__stex_terms_invoke_text:n
3142     \fi: { #1 }
3143   }{
3144     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3145   }
3146 }
3147
3148 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3149   \peek_charcode_remove:NTF ! {
3150     \__stex_terms_invoke_op_custom:nn {#1}
3151   }{
3152     \__stex_terms_invoke_custom:nn {#1}
3153   }
3154 }
3155
3156 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3157   \peek_charcode_remove:NTF ! {
3158     % operator
3159     \peek_charcode_remove:NTF * {
3160       % custom op
3161       \__stex_terms_invoke_op_custom:nn {#1}
3162     }{
3163       % op notation
3164       \peek_charcode:NTF [ {
3165         \__stex_terms_invoke_op_notation:nw {#1}
3166       }{
3167         \__stex_terms_invoke_op_notation:nw {#1}[]
3168       }
3169     }
3170   }{
3171     \peek_charcode_remove:NTF * {
3172       \__stex_terms_invoke_custom:nn {#1}
3173       % custom
3174     }{
3175       % normal
3176       \peek_charcode:NTF [ {
3177         \__stex_terms_invoke_notation:nw {#1}
3178       }{

```

```

3179         \__stex_terms_invoke_notation:nw {#1}[]
3180     }
3181 }
3182 }
3183 }
3184
3185
3186 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3187     \exp_args:Nnx \use:nn {
3188         \def\comp{\_comp}
3189         \str_set:Nn \l_stex_current_symbol_str { #1 }
3190         \bool_set_false:N \l_stex_allow_semantic_bool
3191         \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3192             \comp{ #2 }
3193         }
3194     }{
3195         \stex_reset:N \comp
3196         \stex_reset:N \l_stex_current_symbol_str
3197         \bool_set_true:N \l_stex_allow_semantic_bool
3198     }
3199 }
3200
3201 \keys_define:nn { stex / terms } {
3202     % lang .tl_set_x:N = \l_stex_notation_lang_str ,
3203     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3204     unknown .code:n = \str_set:Nx
3205         \l_stex_notation_variant_str \l_keys_key_str
3206 }
3207
3208 \cs_new_protected:Nn \__stex_terms_args:n {
3209     % \str_clear:N \l_stex_notation_lang_str
3210     \str_clear:N \l_stex_notation_variant_str
3211
3212     \keys_set:nn { stex / terms } { #1 }
3213 }
3214
3215 \cs_new_protected:Nn \stex_find_notation:nn {
3216     \__stex_terms_args:n { #2 }
3217     \seq_if_empty:cTF {
3218         l_stex_symdecl_ #1 _notations
3219     } {
3220         \msg_error:nxxx{stex}{error/nonotation}{#1}{s}
3221     } {
3222         \str_if_empty:NTF \l_stex_notation_variant_str {
3223             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3224         }{
3225             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3226                 \l_stex_notation_variant_str
3227             }{
3228                 % \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3229             }{
3230                 \msg_error:nxxx{stex}{error/nonotation}{#1}{
3231                     ~\l_stex_notation_variant_str
3232                 }

```

```

3233     }
3234   }
3235 }
3236 }
3237
3238 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3239   \exp_args:Nnx \use:nn {
3240     \def\comp{\_comp}
3241     \str_set:Nn \l_stex_current_symbol_str { #1 }
3242     \stex_find_notation:nn { #1 }{ #2 }
3243     \bool_set_false:N \l_stex_allow_semantic_bool
3244     \cs_if_exist:cTF {
3245       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3246     }{
3247       \_stex_term_oms:nnn { #1 }{
3248         #1 \c_hash_str \l_stex_notation_variant_str
3249       }{
3250         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3251       }
3252     }{
3253       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3254         \cs_if_exist:cTF {
3255           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3256         }{
3257           \tl_set:Nx \stex_symbol_after_invokation_tl {
3258             \_stex_reset:N \comp
3259             \_stex_reset:N \stex_symbol_after_invokation_tl
3260             \_stex_reset:N \l_stex_current_symbol_str
3261             \bool_set_true:N \l_stex_allow_semantic_bool
3262           }
3263           \def\comp{\_comp}
3264           \str_set:Nn \l_stex_current_symbol_str { #1 }
3265           \bool_set_false:N \l_stex_allow_semantic_bool
3266           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3267         }{
3268           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3269             ~\l_stex_notation_variant_str
3270           }
3271         }
3272       }{
3273         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3274       }
3275     }
3276   }{
3277     \_stex_reset:N \comp
3278     \_stex_reset:N \l_stex_current_symbol_str
3279     \bool_set_true:N \l_stex_allow_semantic_bool
3280   }
3281 }
3282
3283 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3284   \stex_find_notation:nn { #1 }{ #2 }
3285   \cs_if_exist:cTF {
3286     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs

```

```

3287 }{
3288   \tl_set:Nx \stex_symbol_after_invokation_tl {
3289     \_stex_reset:N \comp
3290     \_stex_reset:N \stex_symbol_after_invokation_tl
3291     \_stex_reset:N \l_stex_current_symbol_str
3292     \bool_set_true:N \l_stex_allow_semantic_bool
3293   }
3294   \def\comp{\_comp}
3295   \str_set:Nn \l_stex_current_symbol_str { #1 }
3296   \bool_set_false:N \l_stex_allow_semantic_bool
3297   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3298 }{
3299   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3300     ~\l_stex_notation_variant_str
3301   }
3302 }
3303 }
3304
3305 \prop_new:N \l__stex_terms_custom_args_prop
3306
3307 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3308   \exp_args:Nnx \use:nn {
3309     \bool_set_false:N \l_stex_allow_semantic_bool
3310     \def\comp{\_comp}
3311     \str_set:Nn \l_stex_current_symbol_str { #1 }
3312     \prop_clear:N \l__stex_terms_custom_args_prop
3313     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3314     \prop_get:cnN {
3315       l_stex_symdecl_#1 _prop
3316     }{ args } \l_tmpa_str
3317     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3318     \tl_set:Nn \arg { \__stex_terms_arg: }
3319     \str_if_empty:NTF \l_tmpa_str {
3320       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3321     }{
3322       \str_if_in:NnTF \l_tmpa_str b {
3323         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3324       }{
3325         \str_if_in:NnTF \l_tmpa_str B {
3326           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3327         }{
3328           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3329         }
3330       }
3331     }
3332     % TODO check that all arguments exist
3333   }{
3334     \_stex_reset:N \l_stex_current_symbol_str
3335     \_stex_reset:N \arg
3336     \_stex_reset:N \comp
3337     \_stex_reset:N \l__stex_terms_custom_args_prop
3338     \bool_set_true:N \l_stex_allow_semantic_bool
3339   }
3340 }

```

```

3341
3342 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3343   \tl_if_empty:nTF {#2}{
3344     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3345     \bool_set_true:N \l_tmpa_bool
3346     \bool_do_while:Nn \l_tmpa_bool {
3347       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int
3348         \int_incr:N \l_tmpa_int
3349       }{
3350         \bool_set_false:N \l_tmpa_bool
3351       }
3352     }
3353   }{
3354     \int_set:Nn \l_tmpa_int { #2 }
3355   }
3356   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3357   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3358     \msg_error:nnxxx{stex}{error/overarity}
3359     {\int_use:N \l_tmpa_int}
3360     {\l_stex_current_symbol_str}
3361     {\str_count:N \l_tmpa_str}
3362   }
3363   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3364   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3365     \bool_lazy_any:nF {
3366       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3367       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3368     }{
3369       \msg_error:nnxx{stex}{error/doubleargument}
3370       {\int_use:N \l_tmpa_int}
3371       {\l_stex_current_symbol_str}
3372     }
3373   }
3374   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3375   \bool_set_true:N \l_stex_allow_semantic_bool
3376   \IfBooleanTF#1{
3377     \stex_annotate_invisible:n { %TODO
3378       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3379     }
3380   }{ %TODO
3381     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3382   }
3383   \bool_set_false:N \l_stex_allow_semantic_bool
3384 }
3385
3386
3387 \cs_new_protected:Nn \_stex_term_arg:nn {
3388   \bool_set_true:N \l_stex_allow_semantic_bool
3389   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3390   \bool_set_false:N \l_stex_allow_semantic_bool
3391 }
3392
3393 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3394   \exp_args:Nnx \use:nn

```

```

3395 { \int_set:Nn \l__stex_terms_downprec { #2 }
3396   \stex_term_arg:nn { #1 }{ #3 }
3397 }
3398 { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3399 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

3400 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
3401   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3402   \tl_set:Nn \l_tmpb_tl {\stex_term_math_arg:nnn{#1}{#2}}
3403   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 } }{
3404     \expandafter\if\expandafter\relax\noexpand#3
3405     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_maybe_sequence:Nn#3{#1}}
3406   }else
3407     \tl_set:Nn \l_tmpa_tl {\__stex_terms_math_assoc_arg_simple:nn{#1}{#3}}
3408   \fi
3409   \l_tmpa_tl
3410 }{
3411   \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3412 }
3413 }
3414
3415 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:Nn {
3416   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3417   \str_if_empty:NTF \l_tmpa_str {
3418     \exp_args:Nx \cs_if_eq:NNTF {
3419       \tl_head:N #1
3420     } \stex_invoke_sequence:n {
3421       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3422       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3423       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3424       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3425       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3426         \exp_not:n{\exp_args:Nnx \use:nn} {
3427           \exp_not:n {
3428             \def\comp{\_varcomp}
3429             \str_set:Nn \l_stex_current_symbol_str
3430             } {varseq://\l_tmpa_str}
3431             \exp_not:n{ ##1 }
3432           }{
3433             \exp_not:n {
3434               \stex_reset:N \comp
3435               \stex_reset:N \l_stex_current_symbol_str
3436             }
3437           }
3438         }}}
3439       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3440       \seq_reverse:N \l_tmpa_seq
3441       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3442       \seq_map_inline:Nn \l_tmpa_seq {
3443         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3444           \exp_args:Nno

```

```

3445         \l_tmpa_cs { ##1 } \l_tmpa_tl
3446     }
3447 }
3448 \tl_set:Nx \l_tmpa_tl {
3449     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3450         \exp_args:No \exp_not:n \l_tmpa_tl
3451     }
3452 }
3453 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3454 }{
3455     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3456 }
3457 } {
3458     \stex_terms_math_assoc_arg_simple:nn{#2} { #1 }
3459 }
3460
3461 }
3462
3463 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3464     \clist_set:Nn \l_tmpa_clist{ #2 }
3465     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3466         \tl_set:Nn \l_tmpa_tl { \stex_term_arg:nn{A#1}{ #2 } }
3467     }{
3468         \clist_reverse:N \l_tmpa_clist
3469         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3470         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3471             \exp_args:No \exp_not:n \l_tmpa_tl
3472         }}
3473         \clist_map_inline:Nn \l_tmpa_clist {
3474             \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3475                 \exp_args:Nno
3476                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3477             }
3478         }
3479     }
3480     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3481 }

```

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

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3482 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3483 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3484 \int_new:N \l__stex_terms_downprec
3485 \int_set_eq:NN \l__stex_terms_downprec \infprec

```

(End definition for `\infprec`, `\neginfprec`, and `\l__stex_terms_downprec`. These variables are documented on page 80.)

Bracketing:


```

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
3486 \tl_set:Nn \l__stex_terms_left_bracket_str (
3487 \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

3488 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3489   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3490     \bool_set_false:N \l__stex_terms_brackets_done_bool
3491     #2
3492   } {
3493     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3494       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3495         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3496         \dobrackets { #2 }
3497       }
3498     }{ #2 }
3499   }
3500 }

(End definition for \__stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

3501 \bool_new:N \l__stex_terms_brackets_done_bool
3502 %\RequirePackage{scalerel}
3503 \cs_new_protected:Npn \dobrackets #1 {
3504   %\ThisStyle{\if D\m@switch
3505   %   \exp_args:Nnx \use:nn
3506   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3507   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3508   % \else
3509   \exp_args:Nnx \use:nn
3510   {
3511     \bool_set_true:N \l__stex_terms_brackets_done_bool
3512     \int_set:Nn \l__stex_terms_downprec \infprec
3513     \l__stex_terms_left_bracket_str
3514     #1
3515   }
3516   {
3517     \bool_set_false:N \l__stex_terms_brackets_done_bool
3518     \l__stex_terms_right_bracket_str
3519     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3520   }
3521   %\fi}
3522 }

```

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

\withbrackets

```

3523 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3524   \exp_args:Nnx \use:nn
3525   {
3526     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }

```

```

3527 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3528 #3
3529 }
3530 {
3531 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3532 {\l__stex_terms_left_bracket_str}
3533 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3534 {\l__stex_terms_right_bracket_str}
3535 }
3536 }

```

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

`\STEXinvisible`

```

3537 \cs_new_protected:Npn \STEXinvisible #1 {
3538 \stex_annotate_invisible:n { #1 }
3539 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3540 \cs_new_protected:Nn \_stex_term_oms:nnn {
3541 \stex_annotate:nnn{ OMID }{ #2 }{
3542 #3
3543 }
3544 }
3545
3546 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3547 \_stex_terms_maybe_brackets:nn { #3 }{
3548 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3549 }
3550 }

```

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

`_stex_term_math_omv:nn`

```

3551 \cs_new_protected:Nn \_stex_term_omv:nn {
3552 \stex_annotate:nnn{ OMV }{ #1 }{
3553 #2
3554 }
3555 }

```

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

`_stex_term_math_oma:nnnn`

```

3556 \cs_new_protected:Nn \_stex_term_oma:nnn {
3557 \stex_annotate:nnn{ OMA }{ #2 }{
3558 #3
3559 }
3560 }
3561
3562 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3563 \_stex_terms_maybe_brackets:nn { #3 }{
3564 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

3565 }
3566 }

```

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

`_stex_term_math_omb:nnnn`

```

3567 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3568   \stex_annotate:nnn{ OMBIND }{ #2 }{
3569     #3
3570   }
3571 }
3572
3573 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3574   \__stex_terms_maybe_brackets:nn { #3 }{
3575     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3576   }
3577 }

```

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

`\symref`
`\symname`

```

3578 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3579
3580 \keys_define:nn { stex / symname } {
3581   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3582   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3583   root     .tl_set_x:N = \l__stex_terms_root_tl
3584 }
3585
3586 \cs_new_protected:Nn \stex_symname_args:n {
3587   \tl_clear:N \l__stex_terms_post_tl
3588   \tl_clear:N \l__stex_terms_pre_tl
3589   \tl_clear:N \l__stex_terms_root_str
3590   \keys_set:nn { stex / symname } { #1 }
3591 }
3592
3593 \NewDocumentCommand \symref { m m }{
3594   \let\compemph_uri_prev:\compemph@uri
3595   \let\compemph@uri\symrefemph@uri
3596   \STEXsymbol{#1}!{ #2 }
3597   \let\compemph@uri\compemph_uri_prev:
3598 }
3599
3600 \NewDocumentCommand \synonym { 0{} m m }{
3601   \stex_symname_args:n { #1 }
3602   \let\compemph_uri_prev:\compemph@uri
3603   \let\compemph@uri\symrefemph@uri
3604   % TODO
3605   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3606   \let\compemph@uri\compemph_uri_prev:
3607 }
3608
3609 \NewDocumentCommand \symname { 0{} m }{
3610   \stex_symname_args:n { #1 }
3611   \stex_get_symbol:n { #2 }

```

```

3612 \str_set:Nx \l_tmpa_str {
3613   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3614 }
3615 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3616
3617 \let\compemph_uri_prev:\compemph@uri
3618 \let\compemph@uri\symrefemph@uri
3619 \exp_args:NNx \use:nn
3620 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3621   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3622 } }
3623 \let\compemph@uri\compemph_uri_prev:
3624 }
3625
3626 \NewDocumentCommand \Symname { 0{} m }{
3627   \stex_symname_args:n { #1 }
3628   \stex_get_symbol:n { #2 }
3629   \str_set:Nx \l_tmpa_str {
3630     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3631   }
3632   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3633   \let\compemph_uri_prev:\compemph@uri
3634   \let\compemph@uri\symrefemph@uri
3635   \exp_args:NNx \use:nn
3636   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\ifmode*\fi{
3637     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3638     \l__stex_terms_post_tl
3639   } }
3640   \let\compemph@uri\compemph_uri_prev:
3641 }

```

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

30.3 Notation Components

```

3642 <@@=stex_notationcomps>

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

3643 \cs_new_protected:Npn \_comp #1 {
3644   \str_if_empty:NF \l_stex_current_symbol_str {
3645     \stex_html_backend:TF {
3646       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3647     }{
3648       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3649     }
3650   }
3651 }
3652
3653 \cs_new_protected:Npn \_varcomp #1 {
3654   \str_if_empty:NF \l_stex_current_symbol_str {
3655     \stex_html_backend:TF {
3656       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3657     }{
3658       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3659     }
3660   }
3661 }
3662
3663 \def\comp{\_comp}
3664
3665 \cs_new_protected:Npn \compemph@uri #1 #2 {
3666   \compemph{ #1 }
3667 }
3668
3669
3670 \cs_new_protected:Npn \compemph #1 {
3671   #1
3672 }
3673
3674 \cs_new_protected:Npn \defemph@uri #1 #2 {
3675   \defemph{#1}
3676 }
3677
3678 \cs_new_protected:Npn \defemph #1 {
3679   \textbf{#1}
3680 }
3681
3682 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3683   \symrefemph{#1}
3684 }
3685
3686 \cs_new_protected:Npn \symrefemph #1 {
3687   \emph{#1}
3688 }
3689
3690 \cs_new_protected:Npn \varemp@uri #1 #2 {
3691   \varemp{#1}
3692 }
3693
3694 \cs_new_protected:Npn \varemp #1 {
3695   #1
3696 }

```

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

\ellipses

```

3697 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3698 \bool_new:N \l_stex_inparray_bool
3699 \bool_set_false:N \l_stex_inparray_bool
3700 \NewDocumentCommand \parray { m m } {
3701   \begin{group}
3702     \bool_set_true:N \l_stex_inparray_bool
3703     \begin{array}{#1}
3704       #2
3705     \end{array}

```

```

3706 \endgroup
3707 }
3708
3709 \NewDocumentCommand \prmatrix { m } {
3710 \begingroup
3711 \bool_set_true:N \l_stex_inarray_bool
3712 \begin{matrix}
3713 #1
3714 \end{matrix}
3715 \endgroup
3716 }
3717
3718 \def \maybepline {
3719 \bool_if:NT \l_stex_inarray_bool {\hline}
3720 }
3721
3722 \def \parrayline #1 #2 {
3723 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3724 }
3725
3726 \def \pmrow #1 { \parrayline{}{ #1 } }
3727
3728 \def \parraylineh #1 #2 {
3729 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3730 }
3731
3732 \def \parraycell #1 {
3733 #1 \bool_if:NT \l_stex_inarray_bool {&}
3734 }

```

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

30.4 Variables

```

3735 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3736 \cs_new_protected:Nn \stex_invoke_variable:n {
3737 \if_mode_math:
3738 \exp_after:wN \__stex_variables_invoke_math:n
3739 \else:
3740 \exp_after:wN \__stex_variables_invoke_text:n
3741 \fi: {#1}
3742 }
3743
3744 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3745 %TODO
3746 }
3747
3748
3749 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3750 \peek_charcode_remove:NTF ! {
3751 \peek_charcode_remove:NTF ! {
3752 \peek_charcode:NTF [ {

```

```

3753     \__stex_variables_invoke_op_custom:nw
3754   }{
3755     % TODO throw error
3756   }
3757 }{
3758   \__stex_variables_invoke_op:n { #1 }
3759 }
3760 }{
3761   \peek_charcode_remove:NTF * {
3762     \__stex_variables_invoke_text:n { #1 }
3763   }{
3764     \__stex_variables_invoke_math_ii:n { #1 }
3765   }
3766 }
3767 }
3768
3769 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3770   \cs_if_exist:cTF {
3771     stex_var_op_notation_ #1 _cs
3772   }{
3773     \exp_args:Nnx \use:nn {
3774       \def\comp{\_varcomp}
3775       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3776       \_stex_term_omv:nn { var://#1 }{
3777         \use:c{stex_var_op_notation_ #1 _cs }
3778       }
3779     }{
3780       \_stex_reset:N \comp
3781       \_stex_reset:N \l_stex_current_symbol_str
3782     }
3783   }{
3784     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3785       \__stex_variables_invoke_math_ii:n {#1}
3786     }{
3787       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3788     }
3789   }
3790 }
3791
3792 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3793   \cs_if_exist:cTF {
3794     stex_var_notation_#1_cs
3795   }{
3796     \tl_set:Nx \stex_symbol_after_invokation_tl {
3797       \_stex_reset:N \comp
3798       \_stex_reset:N \stex_symbol_after_invokation_tl
3799       \_stex_reset:N \l_stex_current_symbol_str
3800       \bool_set_true:N \l_stex_allow_semantic_bool
3801     }
3802     \def\comp{\_varcomp}
3803     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3804     \bool_set_false:N \l_stex_allow_semantic_bool
3805     \use:c{stex_var_notation_#1_cs}
3806   }{

```

```

3807 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3808 }
3809 }

```

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

30.5 Sequences

```

3810 <@@=stex_sequences>
3811
3812 \cs_new_protected:Nn \stex_invoke_sequence:n {
3813   \peek_charcode_remove:NTF ! {
3814     \stex_term_omv:nn {varseq://#1}{
3815       \exp_args:Nnx \use:nn {
3816         \def\comp{\_varcomp}
3817         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3818         \prop_item:cn{stex_varseq_#1_prop}{notation}
3819       }{
3820         \stex_reset:N \comp
3821         \stex_reset:N \l_stex_current_symbol_str
3822       }
3823     }
3824   }{
3825     \bool_set_false:N \l_stex_allow_semantic_bool
3826     \def\comp{\_varcomp}
3827     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3828     \tl_set:Nx \stex_symbol_after_invokation_tl {
3829       \stex_reset:N \comp
3830       \stex_reset:N \stex_symbol_after_invokation_tl
3831       \stex_reset:N \l_stex_current_symbol_str
3832       \bool_set_true:N \l_stex_allow_semantic_bool
3833     }
3834     \use:c { stex_varseq_#1_cs }
3835   }
3836 }
3837 </package>

```


Chapter 31

STEX -Structural Features Implementation

```
3838 <*package>
3839
3840 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3841
3842 Warnings and error messages
3843 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3844   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3845 }
3846 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3847   Symbol~#1~not~assigned~in~interpretmodule~#2
3848 }
3849 \msg_new:nnn{stex}{error/unknownstructure}{
3850   No~structure~#1~found!
3851 }
3852
3853 \msg_new:nnn{stex}{error/unknownfield}{
3854   No~field~#1~in~instance~#2~found!\#3
3855 }
3856
3857 \msg_new:nnn{stex}{error/keyval}{
3858   Invalid~key=value~pair:#1
3859 }
3860 \msg_new:nnn{stex}{error/instantiate/missing}{
3861   Assignments~missing~in~instantiate:~#1
3862 }
3863 \msg_new:nnn{stex}{error/incompatible}{
3864   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3865 }
3866
```

31.1 Imports with modification

```

3867 <@@=stex_copymodule>
3868 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3869   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3870     \tl_set:Nn \l_tmpa_tl { #1 }
3871     \__stex_copymodule_get_symbol_from_cs:
3872   }{
3873     % argument is a string
3874     % is it a command name?
3875     \cs_if_exist:cTF { #1 }{
3876       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3877       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3878       \str_if_empty:NNTF \l_tmpa_str {
3879         \exp_args:Nx \cs_if_eq:NNTF {
3880           \tl_head:N \l_tmpa_tl
3881         } \stex_invoke_symbol:n {
3882           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3883         }{
3884           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3885         }
3886       } {
3887         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3888       }
3889     }{
3890       % argument is not a command name
3891       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3892       % \l_stex_all_symbols_seq
3893     }
3894   }
3895 }
3896
3897 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3898   \str_set:Nn \l_tmpa_str { #1 }
3899   \bool_set_false:N \l_tmpa_bool
3900   \bool_if:NF \l_tmpa_bool {
3901     \tl_set:Nn \l_tmpa_tl {
3902       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3903     }
3904     \str_set:Nn \l_tmpa_str { #1 }
3905     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3906     \seq_map_inline:Nn #2 {
3907       \str_set:Nn \l_tmpb_str { ##1 }
3908       \str_if_eq:eeT { \l_tmpa_str } {
3909         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3910       } {
3911         \seq_map_break:n {
3912           \tl_set:Nn \l_tmpa_tl {
3913             \str_set:Nn \l_stex_get_symbol_uri_str {
3914               ##1
3915             }
3916           }
3917         }
3918       }

```

```

3919     }
3920     \l_tmpa_tl
3921   }
3922 }
3923
3924 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3925   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3926     { \tl_tail:N \l_tmpa_tl }
3927   \tl_if_single:NTF \l_tmpa_tl {
3928     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3929       \exp_after:wN \str_set:Nn \exp_after:wN
3930         \l_stex_get_symbol_uri_str \l_tmpa_tl
3931       \__stex_copymodule_get_symbol_check:n { #1 }
3932     }{
3933       % TODO
3934       % tail is not a single group
3935     }
3936   }{
3937     % TODO
3938     % tail is not a single group
3939   }
3940 }
3941
3942 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3943   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3944     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3945       :~\seq_use:Nn #1 {,~}
3946     }
3947   }
3948 }
3949
3950 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3951   % import module
3952   \stex_import_module_uri:nn { #1 } { #2 }
3953   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3954   \stex_import_require_module:nnnn
3955     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3956     { \l_stex_import_path_str } { \l_stex_import_name_str }
3957
3958   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3959   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3960
3961   % fields
3962   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3963   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3964     \seq_map_inline:cn {c_stex_module_##1_constants}{
3965       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3966         ##1 ? ####1
3967       }
3968     }
3969   }
3970
3971   % setup prop
3972   \seq_clear:N \l_tmpa_seq

```

```

3973 \exp_args:Nn \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3974   name      = \l_stex_current_copymodule_name_str ,
3975   module    = \l_stex_current_module_str ,
3976   from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3977   includes  = \l_tmpa_seq %,
3978   % fields  = \l_tmpa_seq
3979 }
3980 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3981   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3982 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3983 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3984
3985 \stex_if_do_html:T {
3986   \begin{stex_annotate_env} {#4} {
3987     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3988   }
3989   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3990 }
3991 }
3992
3993 \cs_new_protected:Nn \stex_copymodule_end:n {
3994   % apply to every field
3995   \def \l_tmpa_cs ##1 ##2 {#1}
3996
3997   \tl_clear:N \__stex_copymodule_module_tl
3998   \tl_clear:N \__stex_copymodule_exec_tl
3999
4000   %\prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4001   \seq_clear:N \__stex_copymodule_fields_seq
4002
4003   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4004     \seq_map_inline:cn {c_stex_module_##1_constants}{
4005
4006       \tl_clear:N \__stex_copymodule_curr_symbol_tl % <- wrap in current symbol html
4007       \l_tmpa_cs{##1}{####1}
4008
4009       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4010         \str_set_eq:Nc \__stex_copymodule_curr_name_str {l__stex_copymodule_copymodule_##1?####1_name_str}
4011         \stex_if_do_html:T {
4012           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4013             \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
4014           }
4015         }
4016       }{
4017         \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str /
4018       }
4019
4020       \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4021       \prop_put:Nnx \l_tmpa_prop { name } \__stex_copymodule_curr_name_str
4022       \prop_put:Nnx \l_tmpa_prop { module } \l_stex_current_module_str
4023
4024       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4025         \stex_if_do_html:T {
4026           \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {

```

```

4027         $\stex_annotate_invisible:nnn{definiens}{\exp_after:wN \exp_not:N\csname l__st
4028     }
4029 }
4030 \prop_put:Nnn \l_tmpa_prop { defined } { true }
4031 }
4032
4033 \stex_add_constant_to_current_module:n \__stex_copymodule_curr_name_str
4034 \tl_put_right:Nx \__stex_copymodule_module_tl {
4035     \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_n
4036     \prop_set_from_keyval:cn {
4037         l_stex_symdecl_ \l_stex_current_module_str ? \__stex_copymodule_curr_name_str _prop
4038     }{
4039         \prop_to_keyval:N \l_tmpa_prop
4040     }
4041 }
4042
4043 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4044     \stex_if_do_html:T {
4045         \tl_put_right:Nx \__stex_copymodule_curr_symbol_tl {
4046             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
4047         }
4048     }
4049     \tl_put_right:Nx \__stex_copymodule_module_tl {
4050         \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4051             \stex_invoke_symbol:n {
4052                 \l_stex_current_module_str ? \__stex_copymodule_curr_name_str
4053             }
4054         }
4055     }
4056 }
4057
4058 \seq_put_right:Nx \__stex_copymodule_fields_seq {\l_stex_current_module_str ? \__stex_
4059
4060 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4061     \stex_copy_notations:nn {\l_stex_current_module_str ? \__stex_copymodule_curr_name_s
4062 }
4063
4064 \tl_put_right:Nx \__stex_copymodule_exec_tl {
4065     \stex_if_do_html:TF{
4066         \stex_annotate_invisible:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \e
4067     }{
4068         \exp_after:wN \exp_not:n \exp_after:wN {\__stex_copymodule_curr_symbol_tl}
4069     }
4070 }
4071 }
4072 }
4073
4074
4075 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \__stex_copymodule_fields_seq
4076 \tl_put_left:Nx \__stex_copymodule_module_tl {
4077     \prop_set_from_keyval:cn {
4078         l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _pro
4079 }{
4080     \prop_to_keyval:N \l_stex_current_copymodule_prop

```

```

4081   }
4082 }
4083
4084 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4085   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4086 }
4087
4088 \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4089 \stex_debug:nn{copymodule}{result:\meaning \__stex_copymodule_module_tl}
4090 \stex_debug:nn{copymodule}{output:\meaning \__stex_copymodule_exec_tl}
4091
4092 \__stex_copymodule_exec_tl
4093 \stex_if_do_html:T {
4094   \end{stex_annotate_env}
4095 }
4096 }
4097
4098 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4099   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4100   \stex_deactivate_macro:Nn \symdecl {module~environments}
4101   \stex_deactivate_macro:Nn \symdef {module~environments}
4102   \stex_deactivate_macro:Nn \notation {module~environments}
4103   \stex_reactivate_macro:N \assign
4104   \stex_reactivate_macro:N \renamedekl
4105   \stex_reactivate_macro:N \donotcopy
4106   \stex_smsmode_do:
4107 }{
4108   \stex_copymodule_end:n {}
4109 }
4110
4111 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4112   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4113   \stex_deactivate_macro:Nn \symdecl {module~environments}
4114   \stex_deactivate_macro:Nn \symdef {module~environments}
4115   \stex_deactivate_macro:Nn \notation {module~environments}
4116   \stex_reactivate_macro:N \assign
4117   \stex_reactivate_macro:N \renamedekl
4118   \stex_reactivate_macro:N \donotcopy
4119   \stex_smsmode_do:
4120 }{
4121   \stex_copymodule_end:n {
4122     \tl_if_exist:cF {
4123       l__stex_copymodule_copymodule_##1?##2_def_tl
4124     }{
4125       \str_if_eq:eeF {
4126         \prop_item:cn{
4127           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4128         }{ true }{
4129           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4130             ##1?##2
4131           }{\l_stex_current_copymodule_name_str}
4132         }
4133       }
4134     }

```

```

4135 }
4136
4137 \iffalse \begin{stex_annotate_env} \fi
4138 \NewDocumentEnvironment {realization} { 0 } { m } {
4139   \stex_copymodule_start:nnnn { #1 } { #2 } { #2 } { realize }
4140   \stex_deactivate_macro:Nn \symdecl {module~environments}
4141   \stex_deactivate_macro:Nn \symdef {module~environments}
4142   \stex_deactivate_macro:Nn \notation {module~environments}
4143   \stex_reactivate_macro:N \donotcopy
4144   \stex_reactivate_macro:N \assign
4145   \stex_smsmode_do:
4146 } {
4147   \stex_import_module_uri:nn { #1 } { #2 }
4148   \tl_clear:N \__stex_copymodule_exec_tl
4149   \tl_set:Nx \__stex_copymodule_module_tl {
4150     \stex_import_require_module:nnnn
4151     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4152     { \l_stex_import_path_str } { \l_stex_import_name_str }
4153   }
4154
4155   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4156     \seq_map_inline:cn {c_stex_module_##1_constants}{
4157       \str_set:Nx \__stex_copymodule_curr_name_str { \l_stex_current_copymodule_name_str / #1 }
4158       \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?###1_def_tl}{
4159         \stex_if_do_html:T {
4160           \tl_put_right:Nx \__stex_copymodule_exec_tl {
4161             \stex_annotate_invisible:nnn{assignment} {##1?###1} {
4162               $\stex_annotate_invisible:nnn{definien}{}{\exp_after:wN \exp_not:N\csname l__
4163             }
4164           }
4165         }
4166         \tl_put_right:Nx \__stex_copymodule_module_tl {
4167           \prop_put:cnn {l_stex_symdecl_##1?###1_prop}{ defined }{ true }
4168         }
4169       }
4170     }
4171
4172     \exp_args:No \stex_execute_in_module:n \__stex_copymodule_module_tl
4173
4174     \__stex_copymodule_exec_tl
4175     \stex_if_do_html:T {\end{stex_annotate_env}}
4176   }
4177
4178   \NewDocumentCommand \donotcopy { m } {
4179     \str_clear:N \l_stex_import_name_str
4180     \str_set:Nn \l_tmpa_str { #1 }
4181     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4182     \seq_map_inline:Nn \l_stex_all_modules_seq {
4183       \str_set:Nn \l_tmpb_str { ##1 }
4184       \str_if_eq:eeT { \l_tmpa_str } {
4185         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4186       } {
4187         \seq_map_break:n {
4188           \stex_if_do_html:T {

```

```

4189         \stex_if_smsmode:F {
4190             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4191                 \stex_annotate:nnn{domain}{##1}{}}
4192         }
4193     }
4194 }
4195 \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4196 }
4197 }
4198 \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4199     \str_set:Nn \l_tmpb_str { #####1 }
4200     \str_if_eq:eeT { \l_tmpa_str } {
4201         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4202     } {
4203         \seq_map_break:n {\seq_map_break:n {
4204             \stex_if_do_html:T {
4205                 \stex_if_smsmode:F {
4206                     \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4207                         \stex_annotate:nnn{domain}{
4208                             \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4209                         }{}
4210                     }
4211                 }
4212             }
4213             \str_set:Nx \l_stex_import_name_str {
4214                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4215             }
4216         }}
4217     }
4218 }
4219 }
4220 \str_if_empty:NTF \l_stex_import_name_str {
4221     % TODO throw error
4222 }{
4223     \stex_collect_imports:n {\l_stex_import_name_str }
4224     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4225         \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4226         \seq_map_inline:cn {c_stex_module_###1_constants}{
4227             \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4228             \bool_lazy_any:nT {
4229                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4230                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4231                 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4232             }{
4233                 % TODO throw error
4234             }
4235         }
4236     }
4237     \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4238     \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4239     \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4240 }
4241 \stex_smsmode_do:
4242 }

```



```

4243
4244 \NewDocumentCommand \assign { m m }{
4245   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4246   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4247   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_def_tl}{#2}
4248   \stex_smsmode_do:
4249 }
4250
4251 \keys_define:nn { stex / renamedecl } {
4252   name          .str_set_x:N = \l_stex_renamedecl_name_str
4253 }
4254 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4255   \str_clear:N \l_stex_renamedecl_name_str
4256   \keys_set:nn { stex / renamedecl } { #1 }
4257 }
4258
4259 \NewDocumentCommand \renamedecl { O{} m m }{
4260   \__stex_copymodule_renamedecl_args:n { #1 }
4261   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4262   \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4263   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_macroname_str}{#3}
4264   \str_if_empty:NTF \l_stex_renamedecl_name_str {
4265     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4266       \l_stex_get_symbol_uri_str
4267     } }
4268   } {
4269     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str_name_str}{\l_stex
4270       \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4271       \prop_set_eq:cc {l_stex_symdecl_
4272         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4273         _prop
4274       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_prop}
4275       \seq_set_eq:cc {l_stex_symdecl_
4276         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4277         _notations
4278       }{l_stex_symdecl_ \l_stex_get_symbol_uri_str_notations}
4279       \prop_put:cnx {l_stex_symdecl_
4280         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4281         _prop
4282       }{ name }{ \l_stex_renamedecl_name_str }
4283       \prop_put:cnx {l_stex_symdecl_
4284         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4285         _prop
4286       }{ module }{ \l_stex_current_module_str }
4287       \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4288         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4289       }
4290       \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4291         \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4292       } }
4293     }
4294     \stex_smsmode_do:
4295   }
4296

```

```

4297 \stex_deactivate_macro:Nn \assign {copymodules}
4298 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4299 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4300
4301

```

31.2 The feature environment

structural@feature

```

4302 <@@=stex_features>
4303
4304 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4305   \stex_if_in_module:F {
4306     \msg_set:nnn{stex}{error/nomodule}{
4307       Structural~Feature~has~to~occur~in~a~module:\\
4308       Feature~#2~of~type~#1\\
4309       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4310     }
4311     \msg_error:nn{stex}{error/nomodule}
4312   }
4313
4314   \str_set_eq:NN \l_stex_feature_parent_str \l_stex_current_module_str
4315
4316   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4317
4318   \stex_if_do_html:T {
4319     \begin{stex_annotate_env}{ feature:#1 }{\l_stex_feature_parent_str ? #2 - #1}
4320     \stex_annotate_invisible:nnn{header}{\{ #3 }
4321   }
4322 }{
4323   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4324   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4325   \stex_debug:nn{features}{
4326     Feature: \l_stex_last_feature_str
4327   }
4328   \stex_if_do_html:T {
4329     \end{stex_annotate_env}
4330   }
4331 }

```

31.3 Structure

structure

```

4332 <@@=stex_structures>
4333 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4334   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4335     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4336   }
4337   \prop_gput:cxn{c_stex_module_ \l_stex_current_module_str _structures}
4338   {#1}{#2}
4339 }
4340

```

```

4341 \keys_define:nn { stex / features / structure } {
4342   name          .str_set_x:N = \l__stex_structures_name_str ,
4343 }
4344
4345 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4346   \str_clear:N \l__stex_structures_name_str
4347   \keys_set:nn { stex / features / structure } { #1 }
4348 }
4349
4350 \NewDocumentEnvironment{mathstructure}{m O{}}{
4351   \__stex_structures_structure_args:n { #2 }
4352   \str_if_empty:NT \l__stex_structures_name_str {
4353     \str_set:Nx \l__stex_structures_name_str { #1 }
4354   }
4355   \stex_suppress_html:n {
4356     \exp_args:Nx \stex_symdecl_do:nn {
4357       name = \l__stex_structures_name_str ,
4358       def  = {\STEXsymbol{module-type}}{
4359         \stex_term_math_oms:nnnn {
4360           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4361             { ns } ?
4362           \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4363             { name } / \l__stex_structures_name_str - structure
4364         }{}{0}{}
4365       }}
4366     }{ #1 }
4367   }
4368   \exp_args:Nnnx
4369   \begin{structural_feature_module}{ structure }
4370     { \l__stex_structures_name_str }{}
4371   \stex_smsmode_do:
4372 }{
4373   \end{structural_feature_module}
4374   \stex_reset_up_to_module:n \l_stex_last_feature_str
4375   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4376   \seq_clear:N \l_tmpa_seq
4377   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4378     \seq_map_inline:cn{c_stex_module_##1_constants}{
4379       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4380     }
4381   }
4382   \exp_args:Nnno
4383   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4384   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4385   \stex_add_structure_to_current_module:nn
4386     \l__stex_structures_name_str
4387     \l_stex_last_feature_str
4388
4389   \stex_execute_in_module:x {
4390     \tl_set:cn { #1 }{
4391       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4392     }
4393   }
4394 }

```

```

4395
4396 \cs_new:Nn \stex_invoke_structure:nn {
4397   \stex_invoke_symbol:n { #1?#2 }
4398 }
4399
4400 \cs_new_protected:Nn \stex_get_structure:n {
4401   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4402     \tl_set:Nn \l_tmpa_tl { #1 }
4403     \__stex_structures_get_from_cs:
4404   }{
4405     \cs_if_exist:cTF { #1 }{
4406       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4407       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4408       \str_if_empty:NNTF \l_tmpa_str {
4409         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs } \stex_invoke_structure:nn {
4410           \__stex_structures_get_from_cs:
4411         }{
4412           \__stex_structures_get_from_string:n { #1 }
4413         }
4414       }{
4415         \__stex_structures_get_from_string:n { #1 }
4416       }
4417     }{
4418       \__stex_structures_get_from_string:n { #1 }
4419     }
4420   }
4421 }
4422
4423 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4424   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4425     { \tl_tail:N \l_tmpa_tl }
4426   \str_set:Nx \l_tmpa_str {
4427     \exp_after:wN \use_i:nn \l_tmpa_tl
4428   }
4429   \str_set:Nx \l_tmpb_str {
4430     \exp_after:wN \use_ii:nn \l_tmpa_tl
4431   }
4432   \str_set:Nx \l_stex_get_structure_str {
4433     \l_tmpa_str ? \l_tmpb_str
4434   }
4435   \str_set:Nx \l_stex_get_structure_module_str {
4436     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4437   }
4438 }
4439
4440 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4441   \tl_set:Nn \l_tmpa_tl {
4442     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4443   }
4444   \str_set:Nn \l_tmpa_str { #1 }
4445   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4446
4447   \seq_map_inline:Nn \l_stex_all_modules_seq {
4448     \prop_if_exist:cT {c_stex_module_##1_structures} {

```

```

4449 \prop_map_inline:cn {c_stex_module_##1_structures} {
4450 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4451 \prop_map_break:n{\seq_map_break:n{
4452 \tl_set:Nn \l_tmpa_tl {
4453 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4454 \str_set:Nn \l_stex_get_structure_module_str {####2}
4455 }
4456 }}
4457 }
4458 }
4459 }
4460 }
4461 \l_tmpa_tl
4462 }

```

\instantiate

```

4463
4464 \keys_define:nn { stex / instantiate } {
4465   name .str_set_x:N = \l__stex_structures_name_str
4466 }
4467 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4468 \str_clear:N \l__stex_structures_name_str
4469 \keys_set:nn { stex / instantiate } { #1 }
4470 }
4471
4472 \NewDocumentCommand \instantiate {m O{} m m O{}}{
4473 \beginngroup
4474 \stex_get_structure:n {#3}
4475 \__stex_structures_instantiate_args:n { #2 }
4476 \str_if_empty:NT \l__stex_structures_name_str {
4477 \str_set:Nn \l__stex_structures_name_str { #1 }
4478 }
4479 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4480 \seq_clear:N \l__stex_structures_fields_seq
4481 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4482 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4483 \seq_map_inline:cn {c_stex_module_##1_constants}{
4484 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4485 }
4486 }
4487
4488 \tl_if_empty:nF{#5}{
4489 \seq_set_split:Nnn \l_tmpa_seq , {#5}
4490 \prop_clear:N \l_tmpa_prop
4491 \seq_map_inline:Nn \l_tmpa_seq {
4492 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4493 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4494 \msg_error:nnn{stex}{error/keyval}{##1}
4495 }
4496 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4497 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4498 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4499 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4500 \exp_args:Nxx \str_if_eq:nnF

```

```

4501         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4502         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4503         \msg_error:nnxxxx{stex}{error/incompatible}
4504         {\l__stex_structures_dom_str}
4505         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4506         {\l_stex_get_symbol_uri_str}
4507         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4508     }
4509     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4510 }
4511 }
4512
4513 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4514     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4515     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4516
4517     \stex_add_constant_to_current_module:n {\l_tmpa_str}
4518     \stex_execute_in_module:x {
4519         \prop_set_from_keyval:cn { l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _p
4520         name = \l_tmpa_str ,
4521         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4522         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4523         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4524     }
4525     \seq_clear:c {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notations}
4526 }
4527
4528 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4529     \stex_find_notation:nn{##1}{}
4530     \stex_execute_in_module:x {
4531         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4532     }
4533
4534     \stex_copy_control_sequence:ccN
4535     {stex_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_notation_
4536     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4537     \l_tmpa_tl
4538     \exp_args:No \stex_execute_in_module:n \l_tmpa_tl
4539
4540
4541     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4542         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4543         \stex_execute_in_module:x {
4544             \tl_set:cn
4545             {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str\c_hash_str \l_stex_not
4546             { \exp_args:No \exp_not:n \l_tmpa_cs}
4547         }
4548     }
4549
4550 }
4551
4552 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4553 }
4554

```

```

4555 \stex_execute_in_module:x {
4556   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4557   domain = \l_stex_get_structure_module_str ,
4558   \prop_to_keyval:N \l_tmpa_prop
4559 }
4560 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4561 }
4562 \stex_debug:nn{instantiate}{
4563   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4564   \prop_to_keyval:N \l_tmpa_prop
4565 }
4566 \exp_args:Nxx \stex_symdecl_do:nn {
4567   type={\STEXsymbol{module-type}}{
4568     \stex_term_math_oms:nnnn {
4569       \l_stex_get_structure_module_str
4570     }{}{0}{}
4571   }}
4572 }{\l__stex_structures_name_str}
4573 % {
4574   \str_set:Nx \l_stex_get_symbol_uri_str {\l_stex_current_module_str?\l__stex_structures
4575   \tl_set:Nn \l_stex_notation_after_do_tl {\__stex_notation_final:}
4576   \stex_notation_do:nnnnn{}{}{}{}{\comp{#4}}
4577 % }
4578 %\exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4579 \endgroup
4580 \stex_smsmode_do:\ignorespacesandpars
4581 }
4582
4583 \cs_new_protected:Nn \stex_symbol_or_var:n {
4584   \cs_if_exist:cTF{#1}{
4585     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4586     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4587     \str_if_empty:NTF \l_tmpa_str {
4588       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4589       \stex_invoke_variable:n {
4590         \bool_set_true:N \l_stex_symbol_or_var_bool
4591         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4592         \str_set:Nx \l_stex_get_symbol_uri_str {
4593           \exp_after:wN \use:n \l_tmpa_tl
4594         }
4595       }{
4596         \bool_set_false:N \l_stex_symbol_or_var_bool
4597         \stex_get_symbol:n{#1}
4598       }
4599     }{
4600       \__stex_structures_symbolorvar_from_string:n{ #1 }
4601     }
4602   }{
4603     \__stex_structures_symbolorvar_from_string:n{ #1 }
4604   }
4605 }
4606
4607 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4608   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{

```

```

4609     \bool_set_true:N \l_stex_symbol_or_var_bool
4610     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4611   }{
4612     \bool_set_false:N \l_stex_symbol_or_var_bool
4613     \stex_get_symbol:n{#1}
4614   }
4615 }
4616
4617 \keys_define:nn { stex / varinstantiate } {
4618   name      .str_set_x:N = \l__stex_structures_name_str,
4619   bind      .choices:nn =
4620     {forall,exists}
4621     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4622 }
4623
4624 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4625   \str_clear:N \l__stex_structures_name_str
4626   \str_clear:N \l__stex_structures_bind_str
4627   \keys_set:nn { stex / varinstantiate } { #1 }
4628 }
4629
4630 \NewDocumentCommand \varinstantiate {m O{} m m O{}}{
4631   \beginingroup
4632     \stex_get_structure:n {#3}
4633     \__stex_structures_varinstantiate_args:n { #2 }
4634     \str_if_empty:NT \l__stex_structures_name_str {
4635       \str_set:Nn \l__stex_structures_name_str { #1 }
4636     }
4637     \stex_if_do_html:TF{
4638       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4639     }{\use:n}
4640     {
4641       \stex_if_do_html:T{
4642         \stex_annotate_invisible:nnn{domain}{\l_stex_get_structure_module_str}{}}
4643     }
4644     \seq_clear:N \l__stex_structures_fields_seq
4645     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4646     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4647       \seq_map_inline:cn {c_stex_module_##1_constants}{
4648         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4649       }
4650     }
4651     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4652     \prop_clear:N \l_tmpa_prop
4653     \tl_if_empty:nF {#5} {
4654       \seq_set_split:Nnn \l_tmpa_seq , {#5}
4655       \seq_map_inline:Nn \l_tmpa_seq {
4656         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4657         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4658           \msg_error:nnn{stex}{error/keyval}{##1}
4659         }
4660         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_stru
4661         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4662         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol

```



```

4663 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4664 \stex_if_do_html:T{
4665   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4666 }
4667 \bool_if:NTF \l_stex_symbol_or_var_bool {
4668   \exp_args:Nxx \str_if_eq:nnF
4669     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4670     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4671     \msg_error:nnxxxx{stex}{error/incompatible}
4672     {\l__stex_structures_dom_str}
4673     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4674     {\l_stex_get_symbol_uri_str}
4675     {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4676   }
4677   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n}
4678 }{
4679   \exp_args:Nxx \str_if_eq:nnF
4680     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4681     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4682     \msg_error:nnxxxx{stex}{error/incompatible}
4683     {\l__stex_structures_dom_str}
4684     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4685     {\l_stex_get_symbol_uri_str}
4686     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4687   }
4688   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n}
4689 }
4690 }
4691 }
4692 \tl_gclear:N \g__stex_structures_aftergroup_tl
4693 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4694   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l__stex_structures_
4695   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4696   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4697     \stex_find_notation:nn{##1}{
4698       \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4699         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4700       \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}
4701       \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4702         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4703         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4704         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4705       }
4706     }
4707   }
4708   \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4709     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4710       name = \l_tmpa_str ,
4711       args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4712       arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4713       assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4714     }
4715     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4716     {g__stex_structures_tmpa_\l_tmpa_str _cs}

```

```

4717         \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4718         {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4719     }
4720     \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4721 }
4722 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4723     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4724         domain = \l_stex_get_structure_module_str ,
4725         \prop_to_keyval:N \l_tmpa_prop
4726     }
4727     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4728     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4729         \exp_args:Nnx \exp_not:N \use:nn {
4730             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4731             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4732                 \exp_not:n{
4733                     \_varcomp{#4}
4734                 }
4735             }
4736         }{
4737             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4738         }
4739     }
4740 }
4741 }
4742 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4743 \aftergroup\g__stex_structures_aftergroup_tl
4744 \endgroup
4745 \stex_smsmode_do:\ignorespacesandpars
4746 }
4747
4748 \cs_new_protected:Nn \stex_invoke_instance:n {
4749     \peek_charcode_remove:NTF ! {
4750         \stex_invoke_symbol:n{#1}
4751     }{
4752         \_stex_invoke_instance:nn {#1}
4753     }
4754 }
4755
4756
4757 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4758     \peek_charcode_remove:NTF ! {
4759         \exp_args:Nnx \use:nn {
4760             \def\comp{\_varcomp}
4761             \use:c{l_stex_varinstance_#1_op_tl}
4762         }{
4763             \_stex_reset:N \comp
4764         }
4765     }{
4766         \_stex_invoke_varinstance:nn {#1}
4767     }
4768 }
4769
4770 \cs_new_protected:Nn \_stex_invoke_instance:nn {

```

```

4771 \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4772   \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4773 }{
4774   \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4775   \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4776     \prop_to_keyval:N \l_tmpa_prop
4777   }
4778 }
4779 }
4780
4781 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4782   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4783     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4784     \l_tmpa_tl
4785   }{
4786     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{}
4787   }
4788 }

```

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

`\stex_invoke_structure:nnn`

```

4789 % #1: URI of the instance
4790 % #2: URI of the instantiated module
4791 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4792   \tl_if_empty:nTF{ #3 }{
4793     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4794       c_stex_feature_ #2 _prop
4795     }
4796     \tl_clear:N \l_tmpa_tl
4797     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4798     \seq_map_inline:Nn \l_tmpa_seq {
4799       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4800       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4801       \cs_if_exist:cT {
4802         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4803       }{
4804         \tl_if_empty:NF \l_tmpa_tl {
4805           \tl_put_right:Nn \l_tmpa_tl {,}
4806         }
4807         \tl_put_right:Nx \l_tmpa_tl {
4808           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4809         }
4810       }
4811     }
4812     \exp_args:No \mathstrut \l_tmpa_tl
4813   }{
4814     \stex_invoke_symbol:n{#1/#3}
4815   }
4816 }

```

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

```

4817 </package>

```

Chapter 32

STEX -Statements Implementation

```
4818 <*package>
4819
4820 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4821
4822 <@@=stex_statements>
    Warnings and error messages
4823
\titleemph
4824 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4825 \keys_define:nn {stex / definiendum }{
4826   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4827   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4828   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4829   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4830 }
4831 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4832   \str_clear:N \l__stex_statements_definiendum_root_str
4833   \tl_clear:N \l__stex_statements_definiendum_post_tl
4834   \str_clear:N \l__stex_statements_definiendum_gfa_str
4835   \keys_set:nn { stex / definiendum }{ #1 }
4836 }
4837 \NewDocumentCommand \definiendum { O{} m m } {
4838   \__stex_statements_definiendum_args:n { #1 }
4839   \stex_get_symbol:n { #2 }
4840   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4841   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4842     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4843     \tl_set:Nn \l_tmpa_tl { #3 }
4844   } {
4845     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4846     \tl_set:Nn \l_tmpa_tl {
4847       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4848     }
4849   }
4850 } {
4851   \tl_set:Nn \l_tmpa_tl { #3 }
4852 }
4853
4854 % TODO root
4855 \stex_html_backend:TF {
4856   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4857 } {
4858   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4859 }
4860 }
4861 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4862
4863 \NewDocumentCommand \definame { 0{ } m } {
4864   \__stex_statements_definiendum_args:n { #1 }
4865   % TODO: root
4866   \stex_get_symbol:n { #2 }
4867   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4868   \str_set:Nx \l_tmpa_str {
4869     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4870   }
4871   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4872   \stex_html_backend:TF {
4873     \stex_if_do_html:T {
4874       \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4875         \l_tmpa_str\l__stex_statements_definiendum_post_tl
4876       }
4877     }
4878   } {
4879     \exp_args:Nnx \defemph@uri {
4880       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4881     } { \l_stex_get_symbol_uri_str }
4882   }
4883 }
4884 \stex_deactivate_macro:Nn \definame {definition~environments}
4885
4886 \NewDocumentCommand \Definame { 0{ } m } {
4887   \__stex_statements_definiendum_args:n { #1 }
4888   \stex_get_symbol:n { #2 }
4889   \str_set:Nx \l_tmpa_str {
4890     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4891   }
4892   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}

```

```

4893 \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4894 \stex_html_backend:TF {
4895   \stex_if_do_html:T {
4896     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4897       \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4898     }
4899   }
4900 } {
4901   \exp_args:Nnx \defemph@uri {
4902     \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4903   } { \l_stex_get_symbol_uri_str }
4904 }
4905 }
4906 \stex_deactivate_macro:Nn \Definame {definition-environments}
4907
4908 \NewDocumentCommand \premise { m }{
4909   \stex_annotate:nnn{ premise }{}{ #1 }
4910 }
4911 \NewDocumentCommand \conclusion { m }{
4912   \stex_annotate:nnn{ conclusion }{}{ #1 }
4913 }
4914 \NewDocumentCommand \definiens { 0{} m }{
4915   \str_clear:N \l_stex_get_symbol_uri_str
4916   \tl_if_empty:nF {#1} {
4917     \stex_get_symbol:n { #1 }
4918   }
4919   \str_if_empty:NT \l_stex_get_symbol_uri_str {
4920     \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4921       \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4922     }{
4923       % TODO throw error
4924     }
4925   }
4926   \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
4927     {\l_stex_current_module_str}{
4928       \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4929     }{true}{
4930       \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4931       \exp_args:Nx \stex_add_to_current_module:n {
4932         \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4933       }
4934     }
4935   }
4936   \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4937 }
4938
4939 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4940 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4941 \stex_deactivate_macro:Nn \definiens {definition~environments}
4942

```

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

`sdefinition`

```

4943
4944 \keys_define:nn {stex / sdefinition }{
4945   type      .str_set_x:N = \sdefinitiontype,
4946   id        .str_set_x:N = \sdefinitionid,
4947   name      .str_set_x:N = \sdefinitionname,
4948   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4949   title     .tl_set:N     = \sdefinitiontitle
4950 }
4951 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4952   \str_clear:N \sdefinitiontype
4953   \str_clear:N \sdefinitionid
4954   \str_clear:N \sdefinitionname
4955   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4956   \tl_clear:N \sdefinitiontitle
4957   \keys_set:nn { stex / sdefinition }{ #1 }
4958 }
4959
4960 \NewDocumentEnvironment{sdefinition}{0{}}{
4961   \__stex_statements_sdefinition_args:n{ #1 }
4962   \stex_reactivate_macro:N \definiendum
4963   \stex_reactivate_macro:N \definame
4964   \stex_reactivate_macro:N \Definame
4965   \stex_reactivate_macro:N \premise
4966   \stex_reactivate_macro:N \definiens
4967   \stex_if_smsmode:F{
4968     \seq_clear:N \l_tmpa_seq
4969     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4970       \tl_if_empty:nF{ ##1 }{
4971         \stex_get_symbol:n { ##1 }
4972         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4973           \l_stex_get_symbol_uri_str
4974         }
4975       }
4976     }
4977     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4978     \exp_args:Nnnx
4979     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4980     \str_if_empty:NF \sdefinitiontype {
4981       \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4982     }
4983     \str_if_empty:NF \sdefinitionname {
4984       \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4985     }
4986     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4987     \tl_clear:N \l_tmpa_tl
4988     \clist_map_inline:Nn \l_tmpa_clist {
4989       \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4990         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4991       }
4992     }
4993     \tl_if_empty:NTF \l_tmpa_tl {
4994       \__stex_statements_sdefinition_start:
4995     }{
4996       \l_tmpa_tl

```

```

4997     }
4998   }
4999   \stex_ref_new_doc_target:n \sdefinitionid
5000   \stex_smsmode_do:
5001 }{
5002   \stex_suppress_html:n {
5003     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5004   }
5005   \stex_if_smsmode:F {
5006     \clist_set:No \l_tmpa_clist \sdefinitiontype
5007     \tl_clear:N \l_tmpa_tl
5008     \clist_map_inline:Nn \l_tmpa_clist {
5009       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5010         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5011       }
5012     }
5013     \tl_if_empty:NTF \l_tmpa_tl {
5014       \__stex_statements_sdefinition_end:
5015     }{
5016       \l_tmpa_tl
5017     }
5018     \end{stex_annotate_env}
5019   }
5020 }

```

\stexpatchdefinition

```

5021 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5022   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5023     ~(\sdefinitiontitle)
5024   }~}
5025 }
5026 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5027
5028 \newcommand\stexpatchdefinition[3] [] {
5029   \str_set:Nx \l_tmpa_str{ #1 }
5030   \str_if_empty:NTF \l_tmpa_str {
5031     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5032     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5033   }{
5034     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5035     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5036   }
5037 }

```

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

\inlinedef inline:

```

5038 \keys_define:nn {stex / inlinedef }{
5039   type      .str_set_x:N = \sdefinitiontype,
5040   id        .str_set_x:N = \sdefinitionid,
5041   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5042   name      .str_set_x:N = \sdefinitionname
5043 }
5044 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {

```



```

5045 \str_clear:N \sdefinitiontype
5046 \str_clear:N \sdefinitionid
5047 \str_clear:N \sdefinitionname
5048 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5049 \keys_set:nn { stex / inlinedef }{ #1 }
5050 }
5051 \NewDocumentCommand \inlinedef { 0{} m } {
5052   \begingroup
5053   \__stex_statements_inlinedef_args:n{ #1 }
5054   \stex_reactivate_macro:N \definiendum
5055   \stex_reactivate_macro:N \definame
5056   \stex_reactivate_macro:N \Definame
5057   \stex_reactivate_macro:N \premise
5058   \stex_reactivate_macro:N \definiens
5059   \stex_ref_new_doc_target:n \sdefinitionid
5060   \stex_if_smsmode:TF{\stex_suppress_html:n {
5061     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5062   }}{
5063     \seq_clear:N \l_tmpa_seq
5064     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5065       \tl_if_empty:nF{ ##1 }{
5066         \stex_get_symbol:n { ##1 }
5067         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5068           \l_stex_get_symbol_uri_str
5069         }
5070       }
5071     }
5072     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5073     \exp_args:Nnx
5074     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5075       \str_if_empty:NF \sdefinitiontype {
5076         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5077       }
5078       #2
5079       \str_if_empty:NF \sdefinitionname {
5080         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5081         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5082       }
5083     }
5084   }
5085   \endgroup
5086   \stex_smsmode_do:
5087 }

```

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

32.2 Assertions

sassertion

```

5088
5089 \keys_define:nn {stex / sassertion }{
5090   type      .str_set_x:N = \sassertiontype,
5091   id        .str_set_x:N = \sassertionid,

```

```

5092 title .tl_set:N = \sassertiontitle ,
5093 for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5094 name .str_set_x:N = \sassertionname
5095 }
5096 \cs_new_protected:Nn \l__stex_statements_sassertion_args:n {
5097 \str_clear:N \sassertiontype
5098 \str_clear:N \sassertionid
5099 \str_clear:N \sassertionname
5100 \clist_clear:N \l__stex_statements_sassertion_for_clist
5101 \tl_clear:N \sassertiontitle
5102 \keys_set:nn { stex / sassertion }{ #1 }
5103 }
5104
5105 %\tl_new:N \g__stex_statements_aftergroup_tl
5106
5107 \NewDocumentEnvironment{sassertion}{0{}}{
5108 \l__stex_statements_sassertion_args:n{ #1 }
5109 \stex_reactivate_macro:N \premise
5110 \stex_reactivate_macro:N \conclusion
5111 \stex_if_smsmode:F {
5112 \seq_clear:N \l_tmpa_seq
5113 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5114 \tl_if_empty:nF{ ##1 }{
5115 \stex_get_symbol:n { ##1 }
5116 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5117 \l_stex_get_symbol_uri_str
5118 }
5119 }
5120 }
5121 \exp_args:Nnnx
5122 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5123 \str_if_empty:NF \sassertiontype {
5124 \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5125 }
5126 \str_if_empty:NF \sassertionname {
5127 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5128 }
5129 \clist_set:Nn \l_tmpa_clist \sassertiontype
5130 \tl_clear:N \l_tmpa_tl
5131 \clist_map_inline:Nn \l_tmpa_clist {
5132 \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5133 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5134 }
5135 }
5136 \tl_if_empty:NTF \l_tmpa_tl {
5137 \l__stex_statements_sassertion_start:
5138 }{
5139 \l_tmpa_tl
5140 }
5141 }
5142 \str_if_empty:NTF \sassertionid {
5143 \str_if_empty:NF \sassertionname {
5144 \stex_ref_new_doc_target:n { }
5145 }

```

```

5146 } {
5147   \stex_ref_new_doc_target:n \sassertionid
5148 }
5149 \stex_smsmode_do:
5150 }{
5151   \str_if_empty:NF \sassertionname {
5152     \stex_suppress_html:n{\stex_symdecl_do:nn{ }\sassertionname}}
5153     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5154   }
5155   \stex_if_smsmode:F {
5156     \clist_set:Nn \l_tmpa_clist \sassertiontype
5157     \tl_clear:N \l_tmpa_tl
5158     \clist_map_inline:Nn \l_tmpa_clist {
5159       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5160         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5161       }
5162     }
5163     \tl_if_empty:NTF \l_tmpa_tl {
5164       __stex_statements_sassertion_end:
5165     }{
5166       \l_tmpa_tl
5167     }
5168     \end{stex_annotate_env}
5169   }
5170 }

```

\stexpatchassertion

```

5171
5172 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5173   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5174     (\sassertiontitle)
5175   }~}
5176 }
5177 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5178
5179 \newcommand\stexpatchassertion[3] [] {
5180   \str_set:Nx \l_tmpa_str{ #1 }
5181   \str_if_empty:NTF \l_tmpa_str {
5182     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5183     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5184   }{
5185     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2 }
5186     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5187   }
5188 }

```

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

\inlineass inline:

```

5189 \keys_define:nn {stex / inlineass }{
5190   type      .str_set_x:N = \sassertiontype,
5191   id        .str_set_x:N = \sassertionid,
5192   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5193   name      .str_set_x:N = \sassertionname

```

```

5194 }
5195 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5196   \str_clear:N \sassertiontype
5197   \str_clear:N \sassertionid
5198   \str_clear:N \sassertionname
5199   \clist_clear:N \l__stex_statements_sassertion_for_clist
5200   \keys_set:nn { stex / inlineass }{ #1 }
5201 }
5202 \NewDocumentCommand \inlineass { 0{} m } {
5203   \begingroup
5204     \stex_reactivate_macro:N \premise
5205     \stex_reactivate_macro:N \conclusion
5206     \__stex_statements_inlineass_args:n{ #1 }
5207     \str_if_empty:NTF \sassertionid {
5208       \str_if_empty:NF \sassertionname {
5209         \stex_ref_new_doc_target:n {}
5210       }
5211     } {
5212       \stex_ref_new_doc_target:n \sassertionid
5213     }
5214
5215     \stex_if_smsmode:TF{
5216       \str_if_empty:NF \sassertionname {
5217         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5218         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5219       }
5220     }{
5221       \seq_clear:N \l_tmpa_seq
5222       \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5223         \tl_if_empty:nF{ ##1 }{
5224           \stex_get_symbol:n { ##1 }
5225           \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5226             \l_stex_get_symbol_uri_str
5227           }
5228         }
5229       }
5230       \exp_args:Nnx
5231       \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5232         \str_if_empty:NF \sassertiontype {
5233           \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5234         }
5235         #2
5236         \str_if_empty:NF \sassertionname {
5237           \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5238           \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5239           \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5240         }
5241       }
5242     }
5243   \endgroup
5244   \stex_smsmode_do:
5245 }

```

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

32.3 Examples

sexample

```

5246 \keys_define:nn {stex / sexample }{
5247   type      .str_set_x:N = \exampletype,
5248   id        .str_set_x:N = \sexampleid,
5249   title     .tl_set:N     = \sexampletile,
5250   name      .str_set_x:N = \sexamplename ,
5251   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5252 }
5253 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5254   \str_clear:N \sexampletype
5255   \str_clear:N \sexampleid
5256   \str_clear:N \sexamplename
5257   \tl_clear:N \sexampletile
5258   \clist_clear:N \l__stex_statements_sexample_for_clist
5259   \keys_set:nn { stex / sexample }{ #1 }
5260 }
5261
5262 \NewDocumentEnvironment{sexample}{0{}}{
5263   \__stex_statements_sexample_args:n{ #1 }
5264   \stex_reactivate_macro:N \premise
5265   \stex_reactivate_macro:N \conclusion
5266   \stex_if_smsmode:F {
5267     \seq_clear:N \l_tmpa_seq
5268     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5269       \tl_if_empty:NF{ ##1 }{
5270         \stex_get_symbol:n { ##1 }
5271         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5272           \l_stex_get_symbol_uri_str
5273         }
5274       }
5275     }
5276   }
5277   \exp_args:Nnnx
5278   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5279   \str_if_empty:NF \sexampletype {
5280     \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5281   }
5282   \str_if_empty:NF \sexamplename {
5283     \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5284   }
5285   \clist_set:N \l_tmpa_clist \sexampletype
5286   \tl_clear:N \l_tmpa_tl
5287   \clist_map_inline:Nn \l_tmpa_clist {
5288     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5289       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5290     }
5291   }
5292   \tl_if_empty:NTF \l_tmpa_tl {
5293     \__stex_statements_sexample_start:
5294   }{
5295     \l_tmpa_tl
5296   }

```

```

5297 }
5298 \str_if_empty:NF \sexampleid {
5299   \stex_ref_new_doc_target:n \sexampleid
5300 }
5301 \stex_smsmode_do:
5302 ){
5303   \str_if_empty:NF \sexamplename {
5304     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5305   }
5306   \stex_if_smsmode:F {
5307     \clist_set:Nn \l_tmpa_clist \sexamplotype
5308     \tl_clear:N \l_tmpa_tl
5309     \clist_map_inline:Nn \l_tmpa_clist {
5310       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5311         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5312       }
5313     }
5314     \tl_if_empty:NTF \l_tmpa_tl {
5315       \__stex_statements_sexample_end:
5316     }{
5317       \l_tmpa_tl
5318     }
5319     \end{stex_annotate_env}
5320   }
5321 }

```

\stexpatchexample

```

5322
5323 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5324   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5325     (\sexamplotype)
5326   }~}
5327 }
5328 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5329
5330 \newcommand\stexpatchexample[3]{} {
5331   \str_set:Nx \l_tmpa_str{ #1 }
5332   \str_if_empty:NTF \l_tmpa_str {
5333     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5334     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5335   }{
5336     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5337     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5338   }
5339 }

```

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

\inlineex inline:

```

5340 \keys_define:nn {stex / inlineex }{
5341   type      .str_set_x:N = \sexamplotype,
5342   id        .str_set_x:N = \sexampleid,
5343   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5344   name      .str_set_x:N = \sexamplename

```

```

5345 }
5346 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5347   \str_clear:N \sexamplotype
5348   \str_clear:N \sexampleid
5349   \str_clear:N \sexamplename
5350   \clist_clear:N \l__stex_statements_sexample_for_clist
5351   \keys_set:nn { stex / inlineex }{ #1 }
5352 }
5353 \NewDocumentCommand \inlineex { 0{ } m } {
5354   \beginngroup
5355   \stex_reactivate_macro:N \premise
5356   \stex_reactivate_macro:N \conclusion
5357   \__stex_statements_inlineex_args:n{ #1 }
5358   \str_if_empty:NF \sexampleid {
5359     \stex_ref_new_doc_target:n \sexampleid
5360   }
5361   \stex_if_smsmode:TF{
5362     \str_if_empty:NF \sexamplename {
5363       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5364     }
5365   }{
5366     \seq_clear:N \l_tmpa_seq
5367     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5368       \tl_if_empty:nF{ ##1 }{
5369         \stex_get_symbol:n { ##1 }
5370         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5371           \l_stex_get_symbol_uri_str
5372         }
5373       }
5374     }
5375     \exp_args:Nnx
5376     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5377       \str_if_empty:NF \sexamplotype {
5378         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5379       }
5380       #2
5381       \str_if_empty:NF \sexamplename {
5382         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5383         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5384       }
5385     }
5386   }
5387   \endgroup
5388   \stex_smsmode_do:
5389 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5390 \keys_define:nn { stex / sparagraph } {
5391   id          .str_set_x:N    = \sparagraphid ,

```

```

5392 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5393 type .str_set_x:N = \sparapgraphtype ,
5394 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5395 from .tl_set:N = \sparagraphfrom ,
5396 to .tl_set:N = \sparagraphto ,
5397 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5398 name .str_set:N = \sparagraphname ,
5399 imports .tl_set:N = \l__stex_statements_sparagraph_imports_tl
5400 }
5401
5402 \cs_new_protected:Nn \stex_sparagraph_args:n {
5403 \tl_clear:N \l_stex_sparagraph_title_tl
5404 \tl_clear:N \sparagraphfrom
5405 \tl_clear:N \sparagraphto
5406 \tl_clear:N \l_stex_sparagraph_start_tl
5407 \tl_clear:N \l__stex_statements_sparagraph_imports_tl
5408 \str_clear:N \sparagraphid
5409 \str_clear:N \sparapgraphtype
5410 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5411 \str_clear:N \sparagraphname
5412 \keys_set:nn { stex / sparagraph }{ #1 }
5413 }
5414 \newif\if@in@omtext\@in@omtextfalse
5415
5416 \NewDocumentEnvironment {sparagraph} { 0{} } {
5417 \stex_sparagraph_args:n { #1 }
5418 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5419 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5420 }{
5421 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5422 }
5423 \@in@omtexttrue
5424 \stex_if_smsmode:F {
5425 \seq_clear:N \l_tmpa_seq
5426 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5427 \tl_if_empty:NF{ ##1 }{
5428 \stex_get_symbol:n { ##1 }
5429 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5430 \l_stex_get_symbol_uri_str
5431 }
5432 }
5433 }
5434 \exp_args:Nnnx
5435 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5436 \str_if_empty:NF \sparapgraphtype {
5437 \stex_annotate_invisible:nnn{typestrings}{\sparapgraphtype}{ }
5438 }
5439 \str_if_empty:NF \sparagraphfrom {
5440 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5441 }
5442 \str_if_empty:NF \sparagraphto {
5443 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5444 }
5445 \str_if_empty:NF \sparagraphname {

```



```

5446     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5447   }
5448   \clist_set:No \l_tmpa_clist \sparagraphtype
5449   \tl_clear:N \l_tmpa_tl
5450   \clist_map_inline:Nn \sparagraphtype {
5451     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5452       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5453     }
5454   }
5455   \stex_csl_to_imports:No \usemodule \l__stex_statements_sparagraph_imports_tl
5456   \tl_if_empty:NTF \l_tmpa_tl {
5457     \__stex_statements_sparagraph_start:
5458   }{
5459     \l_tmpa_tl
5460   }
5461 }
5462 \clist_set:No \l_tmpa_clist \sparagraphtype
5463 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5464 {
5465   \stex_reactivate_macro:N \definiendum
5466   \stex_reactivate_macro:N \definame
5467   \stex_reactivate_macro:N \Definame
5468   \stex_reactivate_macro:N \premise
5469   \stex_reactivate_macro:N \definiens
5470 }
5471 \str_if_empty:NTF \sparagraphid {
5472   \str_if_empty:NTF \sparagraphname {
5473     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5474       \stex_ref_new_doc_target:n {}
5475     }
5476   } {
5477     \stex_ref_new_doc_target:n {}
5478   }
5479 } {
5480   \stex_ref_new_doc_target:n \sparagraphid
5481 }
5482 \exp_args:NNx
5483 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5484   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5485     \tl_if_empty:nF{ ##1 }{
5486       \stex_get_symbol:n { ##1 }
5487       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5488     }
5489   }
5490 }
5491 \stex_smsmode_do:
5492 \ignorespacesandpars
5493 }{
5494   \str_if_empty:NF \sparagraphname {
5495     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5496     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5497   }
5498   \stex_if_smsmode:F {
5499     \clist_set:No \l_tmpa_clist \sparagraphtype

```

```

5500 \tl_clear:N \l_tmpa_tl
5501 \clist_map_inline:Nn \l_tmpa_clist {
5502   \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5503     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5504   }
5505 }
5506 \tl_if_empty:NTF \l_tmpa_tl {
5507   \__stex_statements_sparagraph_end:
5508 }{
5509   \l_tmpa_tl
5510 }
5511 \end{stex_annotate_env}
5512 }
5513 }

```

\stexpatchparagraph

```

5514
5515 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5516   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5517     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5518       \titleemph{\l_stex_sparagraph_title_tl}:~
5519     }
5520   }{
5521     \titleemph{\l_stex_sparagraph_start_tl}~
5522   }
5523 }
5524 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5525
5526 \newcommand\stexpatchparagraph[3] [] {
5527   \str_set:Nx \l_tmpa_str{ #1 }
5528   \str_if_empty:NTF \l_tmpa_str {
5529     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5530     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5531   }{
5532     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5533     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5534   }
5535 }
5536
5537 \keys_define:nn { stex / inlinepara } {
5538   id      .str_set_x:N = \sparagraphid ,
5539   type    .str_set_x:N = \sparagraphtype ,
5540   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5541   from    .tl_set:N    = \sparagraphfrom ,
5542   to      .tl_set:N    = \sparagraphto ,
5543   name    .str_set:N   = \sparagraphname
5544 }
5545 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5546   \tl_clear:N \sparagraphfrom
5547   \tl_clear:N \sparagraphto
5548   \str_clear:N \sparagraphid
5549   \str_clear:N \sparagraphtype
5550   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5551   \str_clear:N \sparagraphname

```

```

5552 \keys_set:nn { stex / inlinepara }{ #1 }
5553 }
5554 \NewDocumentCommand \inlinepara { 0{} m } {
5555   \beginingroup
5556   \__stex_statements_inlinepara_args:n{ #1 }
5557   \clist_set:No \l_tmpa_clist \sparagraphtype
5558   \str_if_empty:NTF \sparagraphid {
5559     \str_if_empty:NTF \sparagraphname {
5560       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5561         \stex_ref_new_doc_target:n {}
5562       }
5563     } {
5564       \stex_ref_new_doc_target:n {}
5565     }
5566   } {
5567     \stex_ref_new_doc_target:n \sparagraphid
5568   }
5569   \stex_if_smsmode:TF{
5570     \str_if_empty:NF \sparagraphname {
5571       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5572     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5573   }
5574 }{
5575   \seq_clear:N \l_tmpa_seq
5576   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5577     \tl_if_empty:nF{ ##1 }{
5578       \stex_get_symbol:n { ##1 }
5579       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5580         \l_stex_get_symbol_uri_str
5581       }
5582     }
5583   }
5584   \exp_args:Nnx
5585   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5586     \str_if_empty:NF \sparagraphtype {
5587       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5588     }
5589     \str_if_empty:NF \sparagraphfrom {
5590       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5591     }
5592     \str_if_empty:NF \sparagraphto {
5593       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5594     }
5595     \str_if_empty:NF \sparagraphname {
5596       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5597     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5598     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5599   }
5600   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5601     \clist_map_inline:Nn \l_tmpa_seq {
5602       \stex_ref_new_sym_target:n {##1}
5603     }
5604   }
5605   #2

```

```

5606     }
5607   }
5608   \endgroup
5609   \stex_smsmode_do:
5610 }
5611

```

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

```

5612 \endpackage

```

Chapter 33

The Implementation

```
5613 <*package>
5614 <@@=stex_sproof>
5615
5616 %%%%%%%%%%    sproof.dtx    %%%%%%%%%%
5617
```

33.1 Proofs

We first define some keys for the proof environment.

```
5618 \keys_define:nn { stex / spf } {
5619   id          .str_set_x:N = \spfid,
5620   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5621   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5622   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5623   type        .str_set_x:N = \spftype,
5624   title       .tl_set:N    = \spftitle,
5625   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5626   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5627   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5628 }
5629 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5630 \str_clear:N \spfid
5631 \tl_clear:N \l__stex_sproof_spf_for_tl
5632 \tl_clear:N \l__stex_sproof_spf_from_tl
5633 \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5634 \str_clear:N \spftype
5635 \tl_clear:N \spftitle
5636 \tl_clear:N \l__stex_sproof_spf_continues_tl
5637 \tl_clear:N \l__stex_sproof_spf_functions_tl
5638 \tl_clear:N \l__stex_sproof_spf_method_tl
5639 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5640 \keys_set:nn { stex / spf }{ #1 }
5641 }
```

```
\c__stex_sproof_flow_str We define this macro, so that we can test whether the display key has the value flow
5642 \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, \LaTeX 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.

```

5643 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5644 \cs_new_protected:Npn \sproofnumber {
5645   \int_set:Nn \l_tmpa_int {1}
5646   \bool_while_do:nn {
5647     \int_compare_p:nNn {
5648       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5649     } > 0
5650   }{
5651     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5652     \int_incr:N \l_tmpa_int
5653   }
5654 }
5655 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5656   \int_set:Nn \l_tmpa_int {1}
5657   \bool_while_do:nn {
5658     \int_compare_p:nNn {
5659       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5660     } > 0
5661   }{
5662     \int_incr:N \l_tmpa_int
5663   }
5664   \int_compare:nNnF \l_tmpa_int = 1 {
5665     \int_decr:N \l_tmpa_int
5666   }
5667   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5668     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1
5669   }
5670 }
5671
5672 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5673   \int_set:Nn \l_tmpa_int {1}
5674   \bool_while_do:nn {
5675     \int_compare_p:nNn {
5676       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5677     } > 0
5678   }{
5679     \int_incr:N \l_tmpa_int
5680   }
5681   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5682 }
5683
5684 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5685   \int_set:Nn \l_tmpa_int {1}
5686   \bool_while_do:nn {

```

```

5687 \int_compare_p:nNn {
5688 \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5689 } > 0
5690 }{
5691 \int_incr:N \l_tmpa_int
5692 }
5693 \int_decr:N \l_tmpa_int
5694 \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5695 }

```

\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

```

5696 \def\sproof@box{
5697 \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5698 }
5699 \def\sproofend{
5700 \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5701 \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5702 }
5703 }

```

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

spf@*@kw

```

5704 \def\spf@proofsketch@kw{Proof~Sketch}
5705 \def\spf@proof@kw{Proof}
5706 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5707 \AddToHook{begindocument}{
5708 \ltx@ifpackageloaded{babel}{
5709 \makeatletter
5710 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5711 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5712 \input{sproof-ngerman.ldf}
5713 }
5714 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5715 \input{sproof-finnish.ldf}
5716 }
5717 \clist_if_in:NnT \l_tmpa_clist {french}{
5718 \input{sproof-french.ldf}
5719 }
5720 \clist_if_in:NnT \l_tmpa_clist {russian}{
5721 \input{sproof-russian.ldf}
5722 }
5723 \makeatother
5724 }{}
5725 }

```

spfsketch

```

5726 \newcommand\spsketch[2][]{
5727 \begin{group}
5728 \let \premise \stex_proof_premise:

```

```

5729 \__stex_sproof_spf_args:n{#1}
5730 \stex_if_smsmode:TF {
5731   \str_if_empty:NF \spfid {
5732     \stex_ref_new_doc_target:n \spfid
5733   }
5734 }{
5735   \seq_clear:N \l_tmpa_seq
5736   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5737     \tl_if_empty:nF{ ##1 }{
5738       \stex_get_symbol:n { ##1 }
5739       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5740         \l_stex_get_symbol_uri_str
5741       }
5742     }
5743   }
5744   \exp_args:Nnx
5745   \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5746     \str_if_empty:NF \spftype {
5747       \stex_annotate_invisible:nnn{type}{\spftype}{
5748     }
5749     \clist_set:Nn \l_tmpa_clist \spftype
5750     \tl_set:Nn \l_tmpa_tl {
5751       \titleemph{
5752         \tl_if_empty:NTF \spftitle {
5753           \spf@proofsketch@kw
5754         }{
5755           \spftitle
5756         }
5757       }:\sim
5758     }
5759     \clist_map_inline:Nn \l_tmpa_clist {
5760       \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5761         \tl_clear:N \l_tmpa_tl
5762       }
5763     }
5764     \str_if_empty:NF \spfid {
5765       \stex_ref_new_doc_target:n \spfid
5766     }
5767     \l_tmpa_tl #2 \sproofend
5768   }
5769 }
5770 \endgroup
5771 \stex_smsmode_do:
5772 }
5773

```

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

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

```

5774 \newenvironment{spfeq}[2][ ]{
5775   \__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


```

5776 \let \premise \stex_proof_premise:
5777 \stex_if_smsmode:TF {
5778   \str_if_empty:NF \spfid {
5779     \stex_ref_new_doc_target:n \spfid
5780   }
5781 }{
5782   \seq_clear:N \l_tmpa_seq
5783   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5784     \tl_if_empty:nF{ ##1 }{
5785       \stex_get_symbol:n { ##1 }
5786       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5787         \l_stex_get_symbol_uri_str
5788       }
5789     }
5790   }
5791   \exp_args:Nnnx
5792   \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5793   \str_if_empty:NF \spftype {
5794     \stex_annotate_invisible:nnn{type}{\spftype}{}
5795   }
5796
5797   \clist_set:No \l_tmpa_clist \spftype
5798   \tl_clear:N \l_tmpa_tl
5799   \clist_map_inline:Nn \l_tmpa_clist {
5800     \tl_if_exist:cT {\_stex_sproof_spfeq_##1_start:}{
5801       \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_start:}}
5802     }
5803     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5804       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5805     }
5806   }
5807   \tl_if_empty:NTF \l_tmpa_tl {
5808     \_stex_sproof_spfeq_start:
5809   }{
5810     \l_tmpa_tl
5811   }{~#2}
5812   \str_if_empty:NF \spfid {
5813     \stex_ref_new_doc_target:n \spfid
5814   }
5815   \begin{displaymath}\begin{array}{rcll}
5816   }
5817   \stex_smsmode_do:
5818 }{
5819   \stex_if_smsmode:F {
5820     \end{array}\end{displaymath}
5821     \clist_set:No \l_tmpa_clist \spftype
5822     \tl_clear:N \l_tmpa_tl
5823     \clist_map_inline:Nn \l_tmpa_clist {
5824       \tl_if_exist:cT {\_stex_sproof_spfeq_##1_end:}{
5825         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_sproof_spfeq_##1_end:}}
5826       }
5827     }
5828     \tl_if_empty:NTF \l_tmpa_tl {
5829       \_stex_sproof_spfeq_end:

```

```

5830   }{
5831     \l_tmpa_tl
5832   }
5833   \end{stex_annotate_env}
5834 }
5835 }
5836
5837 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5838   \titleemph{
5839     \tl_if_empty:NTF \spftitle {
5840       \spf@proof@kw
5841     }{
5842       \spftitle
5843     }
5844   }:
5845 }
5846 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5847
5848 \newcommand\stexpatchspfeq[3] [] {
5849   \str_set:Nx \l_tmpa_str{ #1 }
5850   \str_if_empty:NTF \l_tmpa_str {
5851     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5852     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5853   }{
5854     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5855     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5856   }
5857 }
5858

```

(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.

```

5859 \newenvironment{sproof}[2] []{
5860   \let \premise \stex_proof_premise:
5861   \intarray_gzero:N \l__stex_sproof_counter_intarray
5862   \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5863   \__stex_sproof_spf_args:n{#1}
5864   \stex_if_smsmode:TF {
5865     \str_if_empty:NF \spfid {
5866       \stex_ref_new_doc_target:n \spfid
5867     }
5868   }{
5869     \seq_clear:N \l_tmpa_seq
5870     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5871       \tl_if_empty:nF{ ##1 }{
5872         \stex_get_symbol:n { ##1 }
5873         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5874           \l_stex_get_symbol_uri_str
5875         }
5876       }
5877     }

```

```

5878 \exp_args:Nnnx
5879 \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5880 \str_if_empty:NF \spftype {
5881   \stex_annotate_invisible:nnn{type}{\spftype}{}
5882 }
5883
5884 \clist_set:No \l_tmpa_clist \spftype
5885 \tl_clear:N \l_tmpa_tl
5886 \clist_map_inline:Nn \l_tmpa_clist {
5887   \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5888     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5889   }
5890   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5891     \tl_set:Nn \l_tmpa_tl {\use:n{}}
5892   }
5893 }
5894 \tl_if_empty:NTF \l_tmpa_tl {
5895   \__stex_sproof_sproof_start:
5896 }{
5897   \l_tmpa_tl
5898 }{~#2}
5899 \str_if_empty:NF \spfid {
5900   \stex_ref_new_doc_target:n \spfid
5901 }
5902 \begin{description}
5903 }
5904 \stex_smsmode_do:
5905 ){
5906   \stex_if_smsmode:F{
5907     \end{description}
5908     \clist_set:No \l_tmpa_clist \spftype
5909     \tl_clear:N \l_tmpa_tl
5910     \clist_map_inline:Nn \l_tmpa_clist {
5911       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5912         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5913       }
5914     }
5915     \tl_if_empty:NTF \l_tmpa_tl {
5916       \__stex_sproof_sproof_end:
5917     }{
5918       \l_tmpa_tl
5919     }
5920     \end{stex_annotate_env}
5921   }
5922 }
5923
5924 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5925   \par\noindent\titleemph{
5926     \tl_if_empty:NTF \spftype {
5927       \spf@proof@kw
5928     }{
5929       \spftype
5930     }
5931   }::

```

```

5932 }
5933 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5934
5935 \newcommand\stexpatchproof[3] [] {
5936   \str_set:Nx \l_tmpa_str{ #1 }
5937   \str_if_empty:NTF \l_tmpa_str {
5938     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5939     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5940   }{
5941     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5942     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5943   }
5944 }

```

\spfidea

```

5945 \newcommand\spfidea[2] []{
5946   \__stex_sproof_spf_args:n{#1}
5947   \titleemph{
5948     \tl_if_empty:NTF \spftype {Proof-Idea}{
5949       \spftype
5950     }:
5951   }~#2
5952   \sproofend
5953 }

```

(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

```

5954 \newenvironment{spfstep}[1] []{
5955   \__stex_sproof_spf_args:n{#1}
5956   \stex_if_smsmode:TF {
5957     \str_if_empty:NF \spfid {
5958       \stex_ref_new_doc_target:n \spfid
5959     }
5960   }{
5961     \@in@omtexttrue
5962     \seq_clear:N \l_tmpa_seq
5963     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5964       \tl_if_empty:nF{ ##1 }{
5965         \stex_get_symbol:n { ##1 }
5966         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5967           \l_stex_get_symbol_uri_str
5968         }
5969       }
5970     }
5971     \exp_args:Nnnx
5972     \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5973     \str_if_empty:NF \spftype {
5974       \stex_annotate_invisible:nnn{type}{\spftype}{ }

```

```

5975 }
5976 \clist_set:No \l_tmpa_clist \spftype
5977 \tl_set:Nn \l_tmpa_tl {
5978   \item[\sproofnumber]
5979   \bool_set_true:N \l__stex_sproof_inc_counter_bool
5980 }
5981 \clist_map_inline:Nn \l_tmpa_clist {
5982   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5983     \tl_clear:N \l_tmpa_tl
5984   }
5985 }
5986 \l_tmpa_tl
5987 \tl_if_empty:NF \spftitle {
5988   {(\titleemph{\spftitle})\enspace}
5989 }
5990 \str_if_empty:NF \spfid {
5991   \stex_ref_new_doc_target:n \spfid
5992 }
5993 }
5994 \stex_smsmode_do:
5995 \ignorespacesandpars
5996 }{
5997   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5998     \__stex_sproof_inc_counter:
5999   }
6000   \stex_if_smsmode:F {
6001     \end{stex_annotate_env}
6002   }
6003 }

```

spfcomment

```

6004 \newenvironment{spfcomment}[1][]{
6005   \__stex_sproof_spf_args:n{#1}
6006   \clist_set:No \l_tmpa_clist \spftype
6007   \tl_set:Nn \l_tmpa_tl {
6008     \item[\sproofnumber]
6009     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6010   }
6011   \clist_map_inline:Nn \l_tmpa_clist {
6012     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6013       \tl_clear:N \l_tmpa_tl
6014     }
6015   }
6016   \l_tmpa_tl
6017 }{
6018   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6019     \__stex_sproof_inc_counter:
6020   }
6021 }

```

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.

```

6022 \newenvironment{subproof}[2][]{
6023   \_stex_sproof_spf_args:n{#1}
6024   \stex_if_smsmode:TF{
6025     \str_if_empty:NF \spfid {
6026       \stex_ref_new_doc_target:n \spfid
6027     }
6028   }{
6029     \seq_clear:N \l_tmpa_seq
6030     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6031       \tl_if_empty:nF{ ##1 }{
6032         \stex_get_symbol:n { ##1 }
6033         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6034           \l_stex_get_symbol_uri_str
6035         }
6036       }
6037     }
6038     \exp_args:Nnnx
6039     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6040     \str_if_empty:NF \spftype {
6041       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6042     }
6043
6044     \clist_set:No \l_tmpa_clist \spftype
6045     \tl_set:Nn \l_tmpa_tl {
6046       \item[\sproofnumber]
6047       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6048     }
6049     \clist_map_inline:Nn \l_tmpa_clist {
6050       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6051         \tl_clear:N \l_tmpa_tl
6052       }
6053     }
6054     \l_tmpa_tl
6055     \tl_if_empty:NF \spftitle {
6056       {(\titleemph{\spftitle})\enspace}
6057     }
6058     {~#2}
6059     \str_if_empty:NF \spfid {
6060       \stex_ref_new_doc_target:n \spfid
6061     }
6062   }
6063   \_stex_sproof_add_counter:
6064   \stex_smsmode_do:
6065 }{
6066   \_stex_sproof_remove_counter:
6067   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6068     \_stex_sproof_inc_counter:
6069   }
6070   \stex_if_smsmode:F{
6071     \end{stex_annotate_env}
6072   }
6073 }

```

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

```

6074 \newenvironment{spfcases}[2] [] {
6075   \tl_if_empty:nTF{#1}{
6076     \begin{subproof}[method=by-cases]{#2}
6077   }{
6078     \begin{subproof}[#1,method=by-cases]{#2}
6079   }
6080 }{
6081   \end{subproof}
6082 }

```

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

```

6083 \newenvironment{spfcase}[2] [] {
6084   \__stex_sproof_spf_args:n{#1}
6085   \stex_if_smsmode:TF {
6086     \str_if_empty:NF \spfid {
6087       \stex_ref_new_doc_target:n \spfid
6088     }
6089   }{
6090     \seq_clear:N \l_tmpa_seq
6091     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6092       \tl_if_empty:nF{ ##1 }{
6093         \stex_get_symbol:n { ##1 }
6094         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6095           \l_stex_get_symbol_uri_str
6096         }
6097       }
6098     }
6099     \exp_args:Nnnx
6100     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6101     \str_if_empty:NF \spftype {
6102       \stex_annotate_invisible:nnn{type}{\spftype}{ }
6103     }
6104     \clist_set:Nn \l_tmpa_clist \spftype
6105     \tl_set:Nn \l_tmpa_tl {
6106       \item[\sproofnumber]
6107       \bool_set_true:N \l__stex_sproof_inc_counter_bool
6108     }
6109     \clist_map_inline:Nn \l_tmpa_clist {
6110       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6111         \tl_clear:N \l_tmpa_tl
6112       }
6113     }
6114     \l_tmpa_tl
6115     \tl_if_empty:nF{#2}{
6116       \titleemph{#2}:~
6117     }
6118   }
6119   \__stex_sproof_add_counter:
6120   \stex_smsmode_do:
6121 }{
6122   \__stex_sproof_remove_counter:
6123   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6124     \__stex_sproof_inc_counter:

```

```

6125 }
6126 \stex_if_smsmode:F{
6127   \clist_set:No \l_tmpa_clist \spftype
6128   \tl_set:Nn \l_tmpa_tl{\sproofend}
6129   \clist_map_inline:Nn \l_tmpa_clist {
6130     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6131       \tl_clear:N \l_tmpa_tl
6132     }
6133   }
6134   \l_tmpa_tl
6135   \end{stex_annotate_env}
6136 }
6137 }

```

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

```

6138 \newcommand\spfcasesketch[3] [] {
6139   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6140 }

```

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.

```

6141 \keys_define:nn { stex / just }{
6142   id      .str_set_x:N = \l__stex_sproof_just_id_str,
6143   method  .tl_set:N   = \l__stex_sproof_just_method_tl,
6144   premises .tl_set:N   = \l__stex_sproof_just_premises_tl,
6145   args    .tl_set:N   = \l__stex_sproof_just_args_tl
6146 }

```

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

\spfjust

```

6147 \newcommand\spfjust[1] [] {}

```

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

\premise

```

6148 \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.

```

6149 \newcommand\justarg[2] [] {#2}
6150 \</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

```
6151 <*package>
6152
6153 %%%%%%%%%% others.dtx %%%%%%%%%%
6154
6155 <@@=stex_others>
        Warnings and error messages
6156 % None

\MSC Math subject classifier

6157 \NewDocumentCommand \MSC {m} {
6158 % TODO
6159 }

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

6160 \@ifpackageloaded{tikzinput}{
6161 \RequirePackage{stex-tikzinput}
6162 }{}
6163
6164 \bool_if:NT \c_stex_persist_mode_bool {
6165 \input{\jobname.sms}
6166 \prop_if_exist:NT\c_stex_mathhub_main_manifest_prop{
6167 \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
6168 \l_tmpa_str
6169 \prop_set_eq:cN { c_stex_mathhub_\l_tmpa_str_manifest_prop }
6170 \c_stex_mathhub_main_manifest_prop
6171 \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
6172 }
6173 }

6174 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6175 <*package>
6176 <@@=stex_modules>
6177
6178 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6179
6180 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX/meta}
6181 \begingroup
6182 \stex_module_setup:nn{
6183   ns=\c_stex_metatheory_ns_str,
6184   meta=NONE
6185 }{Metatheory}
6186 \stex_reactivate_macro:N \symdecl
6187 \stex_reactivate_macro:N \notation
6188 \stex_reactivate_macro:N \symdef
6189 \ExplSyntaxOff
6190 \csname stex_suppress_html:n\endcsname{
6191   % is-a (a:A, a \in A, a is an A, etc.)
6192   \symdecl{isa}[args=ai]
6193   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6194   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6195   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6196
6197   % bind (\forall, \Pi, \lambda etc.)
6198   \symdecl{bind}[args=Bi]
6199   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6200   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6201   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
6202
6203   % implicit bind
6204   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6205
6206   % dummy variable
6207   \symdecl{dummyvar}
6208   \notation{dummyvar}[underscore]{\comp\_}
6209   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6210 \notation{dummyvar}[dash]{\comp{\rm --}}
6211
6212 %fromto (function space, Hom-set, implication etc.)
6213 \symdecl{fromto}[args=ai]
6214 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6215 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6216
6217 % mapto (lambda etc.)
6218 \symdecl{mapto}[args=Bi]
6219 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6220 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6221 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6222
6223 % function/operator application
6224 \symdecl{apply}[args=ia]
6225 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6226 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
6227
6228 % collection of propositions/booleans/truth values
6229 \symdecl{prop}[name=proposition]
6230 \notation{prop}[prop]{\comp{\rm prop}}
6231 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6232
6233 \symdecl{judgmentholds}[args=1]
6234 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6235
6236 % sequences
6237 \symdecl{seqtype}[args=1]
6238 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6239
6240 \symdecl{seqexpr}[args=a]
6241 \notation{seqexpr}[angle,prec=nobrackets]{\comp\langle #1\comp\rangle}{##1\comp,##2}
6242
6243 \symdef{seqmap}[args=abi,setlike]{\comp{\#3 \comp| #2\comp\in \dobrackets{#1} \comp\}}{##1\comp,##2}
6244 \symdef{seqprepend}[args=ia]{#1 \comp{:} #2}{##1 \comp, ##2}
6245 \symdef{seqappend}[args=ai]{#1 \comp{:} #2}{##1 \comp, ##2}
6246 \symdef{seqfoldleft}[args=iabbi]{ \comp{foldl}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6247 \symdef{seqfoldright}[args=iabbi,op=foldr]{ \comp{foldr}\dobrackets{#1,#2}\dobrackets{#3\comp,#4\comp}
6248 \symdef{seqhead}[args=a]{\comp{head}\dobrackets{#1}}{##1 \comp, ##2}
6249 \symdef{seqtail}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6250 \symdef{seqlast}[args=a]{\comp{last}\dobrackets{#1}}{##1 \comp, ##2}
6251 \symdef{seqinit}[args=a]{\comp{tail}\dobrackets{#1}}{##1 \comp, ##2}
6252
6253 \symdef{sequence-index}[args=2,li,prec=nobrackets]{\comp{#1}_{#2}}
6254 \notation{sequence-index}[ui,prec=nobrackets]{\comp{#1}^{#2}}
6255
6256 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{\,\ellipses}}{##1\comp,##2}
6257 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{\,\ellipses,}#2}{##1\comp,##2}
6258 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
6259
6260 % letin ("let", local definitions, variable substitution)
6261 \symdecl{letin}[args=bii]
6262 \notation{letin}[let]{\comp{\rm let}}{\; #1\comp{=}#2\; \comp{\rm in}}{\; #3}
6263 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}

```

```

6264 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6265
6266 % structures
6267 \symdecl*{module-type}[args=1]
6268 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6269 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6270 \notation{mathstruct}[angle,prec=nobrackets]{\comp\langle #1 \comp\rangle}{##1 \comp, ##2}
6271
6272 % objects
6273 \symdecl{object}
6274 \notation{object}{\comp{\mathtt{OBJECT}}}
6275
6276 }
6277
6278 % The following are abbreviations in the sTeX corpus that are left over from earlier
6279 % developments. They will eventually be phased out.
6280
6281 \ExplSyntaxOn
6282 \stex_add_to_current_module:n{
6283   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6284   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6285   \def\livar{\csname sequence-index\endcsname[li]}
6286   \def\uivar{\csname sequence-index\endcsname[ui]}
6287   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6288   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6289 }
6290 \__stex_modules_end_module:
6291 \endgroup
6292 \</package>

```

Chapter 36

Tikzinput Implementation

```
6293 <@@=tikzinput>
6294 <*package>
6295
6296 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6297
6298 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6299 \RequirePackage{l3keys2e}
6300
6301 \keys_define:nn { tikzinput } {
6302   image .bool_set:N = \c_tikzinput_image_bool,
6303   image .default:n = false ,
6304   unknown .code:n = {}
6305 }
6306
6307 \ProcessKeysOptions { tikzinput }
6308
6309 \bool_if:NTF \c_tikzinput_image_bool {
6310   \RequirePackage{graphicx}
6311
6312   \providecommand\usetikzlibrary[]{}
6313   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6314 }{
6315   \RequirePackage{tikz}
6316   \RequirePackage{standalone}
6317
6318   \newcommand \tikzinput [2] [] {
6319     \setkeys{Gin}{#1}
6320     \ifx \Gin@ewidth \Gin@exclamation
6321       \ifx \Gin@eheight \Gin@exclamation
6322         \input { #2 }
6323       \else
6324         \resizebox{!}{ \Gin@eheight }{
6325           \input { #2 }
6326         }
6327       \fi
6328     \else
6329       \ifx \Gin@eheight \Gin@exclamation
6330         \resizebox{ \Gin@ewidth }{!}{
```

```

6331         \input { #2 }
6332     }
6333     \else
6334         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6335             \input { #2 }
6336         }
6337     \fi
6338 \fi
6339 }
6340 }
6341
6342 \newcommand \ctikzinput [2] [] {
6343     \begin{center}
6344         \tikzinput [#1] {#2}
6345     \end{center}
6346 }
6347
6348 \@ifpackageloaded{stex}{
6349     \RequirePackage{stex-tikzinput}
6350 }{}
6351
6352 </package>
6353 <*stex>
6354 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6355 \RequirePackage{stex}
6356 \RequirePackage{tikzinput}
6357
6358 \newcommand\mhtikzinput[2] [] {%
6359     \def\Gin@mhrepos{ }\setkeys{Gin}{#1}%
6360     \stex_in_repository:nn\Gin@mhrepos{
6361         \tikzinput[#1]{\mhp\path{##1}{#2}}
6362     }
6363 }
6364 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
6365
6366 \cs_new_protected:Nn \__tikzinput_usetikzlibrary:nn {
6367     \pgfkeys@spdef\pgf@temp{#1}
6368     \expandafter\ifx\csname tikz@library@\pgf@temp @loaded\endcsname\relax%
6369     \expandafter\global\expandafter\let\csname tikz@library@\pgf@temp @loaded\endcsname=\pgf@temp
6370     \expandafter\edef\csname tikz@library@#1@atcode\endcsname{\the\catcode'\@}
6371     \expandafter\edef\csname tikz@library@#1@barcode\endcsname{\the\catcode'\|}
6372     \expandafter\edef\csname tikz@library@#1@dollarcode\endcsname{\the\catcode'\$}
6373     \catcode'\@=11
6374     \catcode'\|=12
6375     \catcode'\$=3
6376     \pgfutil@InputIfFileExists{#2}{-}{-}
6377     \catcode'\@=\csname tikz@library@#1@atcode\endcsname
6378     \catcode'\|=\csname tikz@library@#1@barcode\endcsname
6379     \catcode'\$=\csname tikz@library@#1@dollarcode\endcsname
6380 }
6381
6382
6383 \newcommand\libusetikzlibrary[1]{

```

```

6384 \prop_if_exist:NF \l_stex_current_repository_prop {
6385   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6386 }
6387 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
6388   \msg_error:nnn{stex}{error/notinarchive}\libusetikzlibrary
6389 }
6390 \seq_clear:N \l__tikzinput_libinput_files_seq
6391 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
6392 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
6393
6394 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
6395   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / tikzlibrary
6396   \IfFileExists{ \l_tmpa_str }{
6397     \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6398   }{}
6399   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
6400   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
6401 }
6402
6403 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / tikzlibrary #1 .code.t
6404 \IfFileExists{ \l_tmpa_str }{
6405   \seq_put_right:No \l__tikzinput_libinput_files_seq \l_tmpa_str
6406 }{}
6407
6408 \seq_if_empty:NTF \l__tikzinput_libinput_files_seq {
6409   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .code.t
6410 }{
6411   \int_compare:nNnTF {\seq_count:N \l__tikzinput_libinput_files_seq} = 1 {
6412     \seq_map_inline:Nn \l__tikzinput_libinput_files_seq {
6413       \__tikzinput_usetikzlibrary:nn{#1}{ ##1 }
6414     }
6415   }{
6416     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusetikzlibrary}{tikzlibrary #1 .co
6417   }
6418 }
6419 }
6420 </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

```
6421 \*package>
6422 \@@=document_structure>
6423 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6424 \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).

```
6425
6426 \keys_define:nn{ document-structure }{
6427   class      .str_set_x:N = \c_document_structure_class_str,
6428   topsect    .str_set_x:N = \c_document_structure_topsect_str,,
6429   unknown     .code:n      = {
6430     \PassOptionsToClass{\CurrentOption}{stex}
6431     \PassOptionsToClass{\CurrentOption}{tikzinput}
6432   }
6433   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6434 }
6435 \ProcessKeysOptions{ document-structure }
6436 \str_if_empty:NT \c_document_structure_class_str {
6437   \str_set:Nn \c_document_structure_class_str {article}
6438 }
6439 \str_if_empty:NT \c_document_structure_topsect_str {
6440   \str_set:Nn \c_document_structure_topsect_str {section}
6441 }
```

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

```
6442 \RequirePackage{xspace}
6443 \RequirePackage{comment}
6444 \RequirePackage{stex}
6445 \AddToHook{begindocument}{
```



```

6446 \ltx@ifpackageloaded{babel}{
6447   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6448   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6449     \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6450   }
6451 }{}
6452 }

```

`\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.

```

6453 \int_new:N \l_document_structure_section_level_int
6454 \str_case:NnF \c_document_structure_topsect_str {
6455   {part}}{
6456     \int_set:Nn \l_document_structure_section_level_int {0}
6457   }
6458   {chapter}{
6459     \int_set:Nn \l_document_structure_section_level_int {1}
6460   }
6461 }{
6462   \str_case:NnF \c_document_structure_class_str {
6463     {book}{
6464       \int_set:Nn \l_document_structure_section_level_int {0}
6465     }
6466     {report}{
6467       \int_set:Nn \l_document_structure_section_level_int {0}
6468     }
6469   }{
6470     \int_set:Nn \l_document_structure_section_level_int {2}
6471   }
6472 }

```

37.2 Document Structure

The structure of the document is given by the `sfragment` environment. The hierarchy is adjusted automatically according to the \LaTeX class in effect.

`\currentsectionlevel` For the `\currentsectionlevel` and `\Currentsectionlevel` macros we use an internal macro `\current@section@level` that only contains the keyword (no markup). We initialize it with “document” as a default. In the generated OMDoc, we only generate a text element of class `omdoc_currentsectionlevel`, which will be instantiated by CSS later.¹⁷

EdN:17

```

6473 \def\current@section@level{document}%
6474 \newcommand\currentsectionlevel{\lowercase\expandafter\current@section@level\space}%
6475 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\space}%

```

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

`\skipfragment`

```

6476 \cs_new_protected:Npn \skipfragment {

```

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

```

6477 \ifcase\l_document_structure_section_level_int
6478 \or\stepcounter{part}
6479 \or\stepcounter{chapter}
6480 \or\stepcounter{section}
6481 \or\stepcounter{subsection}
6482 \or\stepcounter{subsubsection}
6483 \or\stepcounter{paragraph}
6484 \or\stepcounter{subparagraph}
6485 \fi
6486 }

```

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

blindfragment

```

6487 \newcommand\at@begin@blindsfragment[1]{
6488 \newenvironment{blindfragment}
6489 {
6490 \int_incr:N\l_document_structure_section_level_int
6491 \at@begin@blindsfragment\l_document_structure_section_level_int
6492 }{}

```

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

```

6493 \newcommand\sfragment@nonum[2]{
6494 \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6495 \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6496 }

```

(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.

```

6497 \newcommand\sfragment@num[2]{
6498 \tl_if_empty:NTF \l__document_structure_sfragment_short_tl {
6499 \@nameuse{#1}{#2}
6500 }{
6501 \cs_if_exist:NTF\rdfmata@sectioning{
6502 \@nameuse{rdfmata@#1@old}[\l__document_structure_sfragment_short_tl]{#2}
6503 }{
6504 \@nameuse{#1}[\l__document_structure_sfragment_short_tl]{#2}
6505 }
6506 }
6507 %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\sfragment@id
6508 }

```

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

sfragment

```

6509 \keys_define:nn { document-structure / sfragment }{
6510 id .str_set_x:N = \l__document_structure_sfragment_id_str,
6511 date .str_set_x:N = \l__document_structure_sfragment_date_str,

```

```

6512 creators      .clist_set:N = \l__document_structure_sfragment_creators_clist,
6513 contributors  .clist_set:N = \l__document_structure_sfragment_contributors_clist,
6514 srccite       .tl_set:N     = \l__document_structure_sfragment_srccite_tl,
6515 type         .tl_set:N     = \l__document_structure_sfragment_type_tl,
6516 short        .tl_set:N     = \l__document_structure_sfragment_short_tl,
6517 display      .tl_set:N     = \l__document_structure_sfragment_display_tl,
6518 intro       .tl_set:N     = \l__document_structure_sfragment_intro_tl,
6519 imports     .tl_set:N     = \l__document_structure_sfragment_imports_tl,
6520 loadmodules  .bool_set:N   = \l__document_structure_sfragment_loadmodules_bool
6521 }
6522 \cs_new_protected:Nn \l__document_structure_sfragment_args:n {
6523   \str_clear:N \l__document_structure_sfragment_id_str
6524   \str_clear:N \l__document_structure_sfragment_date_str
6525   \clist_clear:N \l__document_structure_sfragment_creators_clist
6526   \clist_clear:N \l__document_structure_sfragment_contributors_clist
6527   \tl_clear:N \l__document_structure_sfragment_srccite_tl
6528   \tl_clear:N \l__document_structure_sfragment_type_tl
6529   \tl_clear:N \l__document_structure_sfragment_short_tl
6530   \tl_clear:N \l__document_structure_sfragment_display_tl
6531   \tl_clear:N \l__document_structure_sfragment_imports_tl
6532   \tl_clear:N \l__document_structure_sfragment_intro_tl
6533   \bool_set_false:N \l__document_structure_sfragment_loadmodules_bool
6534   \keys_set:nn { document-structure / sfragment } { #1 }
6535 }

```

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.

```

6536 \newif@if@mainmatter\@mainmattertrue
6537 \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.

```

6538 \keys_define:nn { document-structure / sectioning }{
6539   name      .str_set_x:N = \l__document_structure_sect_name_str   ,
6540   ref       .str_set_x:N = \l__document_structure_sect_ref_str    ,
6541   clear     .bool_set:N  = \l__document_structure_sect_clear_bool ,
6542   clear     .default:n   = {true}                                ,
6543   num       .bool_set:N  = \l__document_structure_sect_num_bool   ,
6544   num       .default:n   = {true}
6545 }
6546 \cs_new_protected:Nn \l__document_structure_sect_args:n {
6547   \str_clear:N \l__document_structure_sect_name_str
6548   \str_clear:N \l__document_structure_sect_ref_str
6549   \bool_set_false:N \l__document_structure_sect_clear_bool
6550   \bool_set_false:N \l__document_structure_sect_num_bool
6551   \keys_set:nn { document-structure / sectioning } { #1 }
6552 }
6553 \newcommand\omdoc@sectioning[3][]{
6554   \l__document_structure_sect_args:n {#1 }
6555   \let\omdoc@sect@name\l__document_structure_sect_name_str
6556   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6557   \if@mainmatter% numbering not overridden by frontmatter, etc.
6558     \bool_if:NTF \l__document_structure_sect_num_bool {

```

```

6559     \sfragment@num{#2}{#3}
6560   }{
6561     \sfragment@nonum{#2}{#3}
6562   }
6563   \def\current@section@level{\omdoc@sect@name}
6564 \else
6565   \sfragment@nonum{#2}{#3}
6566 \fi
6567 }% 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.

```

6568 \newcommand\sfragment@redefine@addtocontents[1]{%
6569 %\edef\__document_structureimport{#1}%
6570 %\@for\@I:=\__document_structureimport\do{%
6571 %\edef\@path{\csname module@\@I @path\endcsname}%
6572 %\@ifundefined{tf@toc}\relax%
6573 %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6574 %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
6575 %\def\addcontentsline##1##2##3{%
6576 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6577 %\else% hyperref.sty not loaded
6578 %\def\addcontentsline##1##2##3{%
6579 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6580 %\fi
6581 }% hyperref.sty loaded?

```

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

```

6582 \newenvironment{sfragment}[2] []% keys, title
6583 {
6584   \__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.

```

6585 \stex_csl_to_imports:No \usemodule \l__document_structure_sfragment_imports_tl
6586
6587 \bool_if:NT \l__document_structure_sfragment_loadmodules_bool {
6588   \sfragment@redefine@addtocontents{
6589     %\@ifundefined{module@id}\used@modules%
6590     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}
6591   }
6592 }

```

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

```

6593
6594 \stex_document_title:n { #2 }
6595
6596 \int_incr:N\l__document_structure_section_level_int
6597 \ifcase\l__document_structure_section_level_int
6598   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6599   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}

```

```

6600 \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6601 \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6602 \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6603 \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6604 \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6605 \fi
6606 \at@begin@sfragment[#1]\l_document_structure_section_level_int{#2}
6607 \str_if_empty:NF \l__document_structure_sfragment_id_str {
6608   \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
6609 }
6610 }% for customization
6611 {}

```

and finally, we localize the sections

```

6612 \newcommand\omdoc@part@kw{Part}
6613 \newcommand\omdoc@chapter@kw{Chapter}
6614 \newcommand\omdoc@section@kw{Section}
6615 \newcommand\omdoc@subsection@kw{Subsection}
6616 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6617 \newcommand\omdoc@paragraph@kw{paragraph}
6618 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.3 Front and Backmatter

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

`\printindex`

```

6619 \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).

```

6620 \cs_if_exist:NTF\frontmatter{
6621   \let\__document_structure_orig_frontmatter\frontmatter
6622   \let\frontmatter\relax
6623 }{
6624   \tl_set:Nn\__document_structure_orig_frontmatter{
6625     \clearpage
6626     \@mainmatterfalse
6627     \pagenumbering{roman}
6628   }
6629 }
6630 \cs_if_exist:NTF\backmatter{
6631   \let\__document_structure_orig_backmatter\backmatter
6632   \let\backmatter\relax
6633 }{
6634   \tl_set:Nn\__document_structure_orig_backmatter{
6635     \clearpage
6636     \@mainmatterfalse

```

```

6637 \pagenumbering{roman}
6638 }
6639 }

```

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.

```

6640 \newenvironment{frontmatter}{
6641   \_document_structure_orig_frontmatter
6642 }{
6643   \cs_if_exist:NTF\mainmatter{
6644     \mainmatter
6645   }{
6646     \clearpage
6647     \@mainmattertrue
6648     \pagenumbering{arabic}
6649   }
6650 }

```

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

```

6651 \newenvironment{backmatter}{
6652   \_document_structure_orig_backmatter
6653 }{
6654   \cs_if_exist:NTF\mainmatter{
6655     \mainmatter
6656   }{
6657     \clearpage
6658     \@mainmattertrue
6659     \pagenumbering{arabic}
6660   }
6661 }

```

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

```

6662 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6663 \def \c__document_structure_document_str{document}
6664 \newcommand\afterprematurestop{}
6665 \def\prematurestop@endsfragment{
6666   \unless\ifx\@currenvir\c__document_structure_document_str
6667     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6668       \expandafter\prematurestop@endsfragment
6669     }
6670   }
6671 \providecommand\prematurestop{
6672   \message{Stopping~sTeX~processing~prematurely}
6673   \prematurestop@endsfragment
6674   \afterprematurestop
6675   \end{document}
6676 }

```

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

37.4 Global Variables

\setSGvar set a global variable

```
6677 \RequirePackage{etoolbox}
6678 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}
```

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

\useSGvar use a global variable

```
6679 \newrobustcmd\useSGvar[1]{%
6680   \@ifundefined{sTeX@Gvar@#1}
6681   {\PackageError{document-structure}
6682    {The sTeX Global variable #1 is undefined}
6683    {set it with \protect\setSGvar}}
6684   \@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.

```
6685 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6686   \@ifundefined{sTeX@Gvar@#1}
6687   {\PackageError{document-structure}
6688    {The sTeX Global variable #1 is undefined}
6689    {set it with \protect\setSGvar}}
6690   {\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.

```
6691 \*cls)
6692 \@@=notesslides)
6693 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6694 \RequirePackage{13keys2e}
6695
6696 \keys_define:nn{notesslides / cls}{
6697   class .str_set_x:N = \c__notesslides_class_str,
6698   notes .bool_set:N = \c__notesslides_notes_bool ,
6699   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6700   docopt .str_set_x:N = \c__notesslides_docopt_str,
6701   unknown .code:n = {
6702     \PassOptionsToPackage{\CurrentOption}{document-structure}
6703     \PassOptionsToClass{\CurrentOption}{beamer}
6704     \PassOptionsToPackage{\CurrentOption}{notesslides}
6705     \PassOptionsToPackage{\CurrentOption}{stex}
6706   }
6707 }
6708 \ProcessKeysOptions{ notesslides / cls }
6709
6710 \str_if_empty:NF \c__notesslides_class_str {
6711   \PassOptionsToPackage{class=\c__notesslides_class_str}{document-structure}
6712 }
6713
6714 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{book}{
6715   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6716 }
6717 \exp_args:No \str_if_eq:nnT\c__notesslides_class_str{report}{
6718   \PassOptionsToPackage{defaulttopsect=part}{notesslides}
6719 }
6720
6721 \RequirePackage{stex}
```



```

6722 \stex_html_backend:T {
6723   \bool_set_true:N\c__notesslides_notes_bool
6724 }
6725
6726 \bool_if:NTF \c__notesslides_notes_bool {
6727   \PassOptionsToPackage{notes=true}{notesslides}
6728 }{
6729   \PassOptionsToPackage{notes=false}{notesslides}
6730 }
6731 \</cls>

```

now we do the same for the notesslides package.

```

6732 \*package>
6733 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6734 \RequirePackage{13keys2e}
6735
6736 \keys_define:nn{notesslides / pkg}{
6737   topsect      .str_set_x:N = \c__notesslides_topsect_str,
6738   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
6739   notes        .bool_set:N = \c__notesslides_notes_bool ,
6740   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6741   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6742   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
6743   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6744   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
6745   unknown      .code:n      = {
6746     \PassOptionsToClass{\CurrentOption}{stex}
6747     \PassOptionsToClass{\CurrentOption}{tikzinput}
6748   }
6749 }
6750 \ProcessKeysOptions{ notesslides / pkg }
6751
6752 \RequirePackage{stex}
6753 \stex_html_backend:T {
6754   \bool_set_true:N\c__notesslides_notes_bool
6755 }
6756
6757 \newif\ifnotes
6758 \bool_if:NTF \c__notesslides_notes_bool {
6759   \notesttrue
6760 }{
6761   \notesfalse
6762 }
6763

```

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

```

6764 \str_if_empty:NTF \c__notesslides_topsect_str {
6765   \str_set_eq:NN \__notesslidestopsect \c__notesslides_defaulttopsec_str
6766 }{
6767   \str_set_eq:NN \__notesslidestopsect \c__notesslides_topsect_str
6768 }
6769 \PassOptionsToPackage{topsect=\__notesslidestopsect}{document-structure}
6770 \</package>

```

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

```

6771 <*cls>
6772 \bool_if:NTF \c__notesslides_notes_bool {
6773   \str_if_empty:NT \c__notesslides_class_str {
6774     \str_set:Nn \c__notesslides_class_str {article}
6775   }
6776   \exp_after:wN\LoadClass\exp_after:wN[\c__notesslides_docostr]
6777     {\c__notesslides_class_str}
6778 }{
6779   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6780   \newcounter{Item}
6781   \newcounter{paragraph}
6782   \newcounter{subparagraph}
6783   \newcounter{Hfootnote}
6784 }
6785 \RequirePackage{document-structure}

```

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

```

6786 \RequirePackage{notesslides}
6787 </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.

```

6788 <*package>
6789 \bool_if:NT \c__notesslides_notes_bool {
6790   \RequirePackage{a4wide}
6791   \RequirePackage{marginnote}
6792   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6793   \RequirePackage{mdframed}
6794   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6795   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6796 }
6797 \RequirePackage{stex-tikzinput}
6798 \RequirePackage{etoolbox}
6799 \RequirePackage{amssymb}
6800 \RequirePackage{amsmath}
6801 \RequirePackage{comment}
6802 \RequirePackage{textcomp}
6803 \RequirePackage{url}
6804 \RequirePackage{graphicx}
6805 \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

```

6806 \bool_if:NT \c__notesslides_notes_bool {
6807   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
6808 }
6809
6810
6811 \NewDocumentCommand \libusetheme {0{} m} {
6812   \bool_if:NTF \c__notesslides_notes_bool {
6813     \libusepackage[#1]{beamernotestheme#2}
6814   }{
6815     \libusepackage[#1]{beamertheme#2}
6816   }
6817 }

```

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

```

6818 \newcounter{slide}
6819 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6820 \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.

```

6821 \bool_if:NTF \c__notesslides_notes_bool {
6822   \renewenvironment{note}{\ignorespaces}{}
6823 }{
6824   \excludecomment{note}
6825 }

```

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.

```

6826 \bool_if:NT \c__notesslides_notes_bool {
6827   \newlength{\slideframewidth}
6828   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6829 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6830   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6831     \bool_set_true:N #1
6832   }{
6833     \bool_set_false:N #1
6834   }
6835 }
6836 \keys_define:nn{notesslides / frame}{
6837   label .str_set_x:N = \l__notesslides_frame_label_str,
6838   allowframebreaks .code:n = {
6839     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6840   },
6841   allowdisplaybreaks .code:n = {

```

¹⁸EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

```

6842     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_allowdisplaybreaks\_bool { #1 }
6843 },
6844 fragile .code:n = {
6845     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_fragile\_bool { #1 }
6846 },
6847 shrink .code:n = {
6848     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_shrink\_bool { #1 }
6849 },
6850 squeeze .code:n = {
6851     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_squeeze\_bool { #1 }
6852 },
6853 t .code:n = {
6854     \_notesslides\_do\_yes\_param:Nn \l\_notesslides\_frame\_t\_bool { #1 }
6855 },
6856 }
6857 \cs\_new\_protected:Nn \_notesslides\_frame\_args:n {
6858     \str\_clear:N \l\_notesslides\_frame\_label\_str
6859     \bool\_set\_true:N \l\_notesslides\_frame\_allowframebreaks\_bool
6860     \bool\_set\_true:N \l\_notesslides\_frame\_allowdisplaybreaks\_bool
6861     \bool\_set\_true:N \l\_notesslides\_frame\_fragile\_bool
6862     \bool\_set\_true:N \l\_notesslides\_frame\_shrink\_bool
6863     \bool\_set\_true:N \l\_notesslides\_frame\_squeeze\_bool
6864     \bool\_set\_true:N \l\_notesslides\_frame\_t\_bool
6865     \keys\_set:nn { notesslides / frame }{ #1 }
6866 }

```

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

```

6867 \renewenvironment{frame}[1][]{
6868     \_notesslides\_frame\_args:n{#1}
6869     \sffamily
6870     \stepcounter{slide}
6871     \def\@currentlabel{\theslide}
6872     \str\_if\_empty:NF \l\_notesslides\_frame\_label\_str {
6873         \label{\l\_notesslides\_frame\_label\_str}
6874     }

```

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

```

6875 \def\itemize@level{outer}
6876 \def\itemize@outer{outer}
6877 \def\itemize@inner{inner}
6878 \renewcommand\newpage{\addtocounter{framenum}{1}}
6879 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6880 \renewenvironment{itemize}{
6881     \ifx\itemize@level\itemize@outer
6882         \def\itemize@label{\$ \rhd \$}
6883     \fi
6884     \ifx\itemize@level\itemize@inner
6885         \def\itemize@label{\$ \scriptstyle \rhd \$}
6886     \fi
6887     \begin{list}
6888     {\itemize@label}
6889     {\setlength{\labelsep}{.3em}
6890      \setlength{\labelwidth}{.5em}
6891      \setlength{\leftmargin}{1.5em}
6892     }

```

```

6893     \edef\itemize@level{\itemize@inner}
6894   }{
6895     \end{list}
6896   }

```

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

```

6897     \stex_html_backend:TF {
6898       \begin{stex_annotate_env}{frame}{}\vbox\bgroup
6899     }{
6900       \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwid
6901     }
6902   }{
6903     \stex_html_backend:TF {
6904       \miko@slidelabel\egroup\end{stex_annotate_env}
6905     }\medskip\miko@slidelabel\end{mdframed}}
6906   }

```

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

`\frametitle`

```

6907   \renewcommand{\frametitle}[1]{
6908     \stex_document_title:n { #1 }
6909     {\Large\bf\sf\color{blue}{#1}}\medskip
6910   }
6911 }

```

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

EdN:19

`\pause`

```

19
6912 \bool_if:NT \c__notesslides_notes_bool {
6913   \newcommand\pause{}
6914 }

```

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

`nparagraph`

```

6915 \bool_if:NTF \c__notesslides_notes_bool {
6916   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6917 }{
6918   \excludecomment{nparagraph}
6919 }

```

`nfragment`

```

6920 \bool_if:NTF \c__notesslides_notes_bool {
6921   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6922 }{
6923   \excludecomment{nfragment}
6924 }

```

`ndefinition`

```

6925 \bool_if:NTF \c__notesslides_notes_bool {
6926   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6927 }{
6928   \excludecomment{ndefinition}
6929 }

```

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

nassertion

```
6930 \bool_if:NTF \c_notesslides_notes_bool {
6931   \newenvironment{nassertion}[1][\begin{sassertion}[#1]]{\end{sassertion}}
6932 }{
6933   \excludecomment{nassertion}
6934 }
```

nsproof

```
6935 \bool_if:NTF \c_notesslides_notes_bool {
6936   \newenvironment{nproof}[2][\begin{sproof}[#1][#2]]{\end{sproof}}
6937 }{
6938   \excludecomment{nproof}
6939 }
```

nexample

```
6940 \bool_if:NTF \c_notesslides_notes_bool {
6941   \newenvironment{nexample}[1][\begin{sexample}[#1]]{\end{sexample}}
6942 }{
6943   \excludecomment{nexample}
6944 }
```

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

```
6945 \def\inputref@preskip{\smallskip}
6946 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
6947 \let\orig@inputref\inputref
6948 \def\inputref{\@ifstar\ninputref\orig@inputref}
6949 \newcommand\ninputref[2][\{
6950   \bool_if:NT \c_notesslides_notes_bool {
6951     \orig@inputref[#1][#2]
6952   }
6953 }
```

(End definition for \inputref*. This function is documented on page 54.)

38.3 Header and Footer Lines

Now, we set up the infrastructure for the footer line of the slides, we use boxes for the logos, so that they are only loaded once, that considerably speeds up processing.

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

```
6954 \newlength{\slideologoheight}
6955
6956 \bool_if:NTF \c_notesslides_notes_bool {
6957   \setlength{\slideologoheight}{.4cm}
6958 }{
6959   \setlength{\slideologoheight}{1cm}
6960 }
6961 \newsavebox{\slideologo}
```

```

6962 \sbox{\slidelogo}{\TeX}
6963 \newrobustcmd{\setslidelogo}[1]{
6964   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6965 }

```

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

`\setsource` `\source` stores the writer’s name. By default it is *Michael Kohlhase* since he is the main user and designer of this package. `\setsource{<name>}` can change the writer’s name.

```

6966 \def\source{Michael Kohlhase}% customize locally
6967 \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.

```

6968 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6969 \newsavebox{\cclogo}
6970 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6971 \newif\ifcchref\cchreffalse
6972 \AtBeginDocument{
6973   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6974 }
6975 \def\licensing{
6976   \ifcchref
6977     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6978   \else
6979     {\usebox{\cclogo}}
6980   \fi
6981 }
6982 \newrobustcmd{\setlicensing}[2][{}]{
6983   \def\@url{#1}
6984   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6985   \ifx\@url\@empty
6986     \def\licensing{{\usebox{\cclogo}}}
6987   \else
6988     \def\licensing{
6989       \ifcchref
6990         \href{#1}{\usebox{\cclogo}}
6991       \else
6992         {\usebox{\cclogo}}
6993       \fi
6994     }
6995   \fi
6996 }

```

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

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

```

6997 \newrobustcmd\miko@slidelabel{
6998   \vbox to \slidelogoheight{

```

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

```

6999     \vss\hbox to \slidewidth
7000     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7001   }
7002 }

```

(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.

```

7003 \def\Gin@mhrepos{}
7004 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7005 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
7006 \newrobustcmd\frameimage[2][{}{
7007   \stepcounter{slide}
7008   \bool_if:NT \c__notesslides_frameimages_bool {
7009     \def\Gin@ewidth{} \setkeys{Gin}{#1}
7010     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7011     \begin{center}
7012       \bool_if:NTF \c__notesslides_fiboxed_bool {
7013         \fbox{
7014           \ifx\Gin@ewidth\@empty
7015             \ifx\Gin@mhrepos\@empty
7016               \mhgraphics[width=\slidewidth,#1]{#2}
7017             \else
7018               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7019             \fi
7020           \else% Gin@ewidth empty
7021             \ifx\Gin@mhrepos\@empty
7022               \mhgraphics[#1]{#2}
7023             \else
7024               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7025             \fi
7026           \fi% Gin@ewidth empty
7027         }
7028       }{
7029         \ifx\Gin@ewidth\@empty
7030           \ifx\Gin@mhrepos\@empty
7031             \mhgraphics[width=\slidewidth,#1]{#2}
7032           \else
7033             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7034           \fi
7035           \ifx\Gin@mhrepos\@empty
7036             \mhgraphics[#1]{#2}
7037           \else
7038             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7039           \fi
7040         \fi% Gin@ewidth empty
7041       }
7042     \end{center}
7043     \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7044     \bool_if:NF \c__notesslides_notes_bool { \vfill }

```



```

7045 }
7046 } % 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.

```

7047 \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

```

7048 \AddToHook{begindocument}{
7049   \definecolor{green}{rgb}{0,.5,0}
7050   \definecolor{purple}{cmk}{.3,1,0,.17}
7051 }

```

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`.

```

7052 % \def\STpresent#1{\textcolor{blue}{#1}}
7053 \def\defemph#1{\textcolor{magenta}{#1}}
7054 \def\symrefemph#1{\textcolor{cyan}{#1}}
7055 \def\compemph#1{\textcolor{blue}{#1}}
7056 \def\titleemph#1{\textcolor{blue}{#1}}
7057 \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.

```

7058 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7059 \def\smalltextwarning{
7060   \pgfuseimage{miko@small@dbend}
7061   \xspace
7062 }
7063 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7064 \newrobustcmd\textwarning{
7065   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7066   \xspace
7067 }
7068 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7069 \newrobustcmd\bigtextwarning{
7070   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7071   \xspace
7072 }

```

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

```

7073 \newrobustcmd\putgraphicsat[3]{
7074   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7075 }
7076 \newrobustcmd\putat[2]{
7077   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7078 }

```

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`.

```

7079 \bool_if:NT \c__notesslides_sectocframes_bool {
7080   \str_if_eq:VnTF \__notesslidesstopsect{part}{
7081     \newcounter{chapter}\counterwithin*{section}{chapter}
7082   }{
7083     \str_if_eq:VnTF \__notesslidesstopsect{chapter}{
7084       \newcounter{chapter}\counterwithin*{section}{chapter}
7085     }
7086   }
7087 }
```

`\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

7088 \def\part@prefix{}
7089 \@ifpackageloaded{document-structure}{}{
7090   \str_case:VnF \__notesslidesstopsect {
7091     {part}{
7092       \int_set:Nn \l_document_structure_section_level_int {0}
7093       \def\thesection{\arabic{chapter}.\arabic{section}}
7094       \def\part@prefix{\arabic{chapter}.}
7095     }
7096     {chapter}{
7097       \int_set:Nn \l_document_structure_section_level_int {1}
7098       \def\thesection{\arabic{chapter}.\arabic{section}}
7099       \def\part@prefix{\arabic{chapter}.}
7100     }
7101   }{
7102     \int_set:Nn \l_document_structure_section_level_int {2}
7103     \def\part@prefix{}
7104   }
7105 }
7106
7107 \bool_if:NF \c__notesslides_notes_bool { % only in slides
```

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

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

sfragment

```

7108 \renewenvironment{sfragment}[2][]{
7109   \__document_structure_sfragment_args:n { #1 }
7110   \int_incr:N \l_document_structure_section_level_int
7111   \bool_if:NT \c__notesslides_sectocframes_bool {
7112     \stepcounter{slide}
7113     \begin{frame}[noframenumbering]
7114       \vfill\Large\centering
7115       \red{
7116         \ifcase\l_document_structure_section_level_int\or
```

```

7117         \stepcounter{part}
7118         \def\__notesslideslabel{\{ \omdoc@part@kw\}~\Roman{part}}
7119         \def\currentsectionlevel{\omdoc@part@kw}
7120     \or
7121         \stepcounter{chapter}
7122         \def\__notesslideslabel{\{ \omdoc@chapter@kw\}~\arabic{chapter}}
7123         \def\currentsectionlevel{\omdoc@chapter@kw}
7124     \or
7125         \stepcounter{section}
7126         \def\__notesslideslabel{\part@prefix\arabic{section}}
7127         \def\currentsectionlevel{\omdoc@section@kw}
7128     \or
7129         \stepcounter{subsection}
7130         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7131         \def\currentsectionlevel{\omdoc@subsection@kw}
7132     \or
7133         \stepcounter{subsubsection}
7134         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7135         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7136     \or
7137         \stepcounter{paragraph}
7138         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7139         \def\currentsectionlevel{\omdoc@paragraph@kw}
7140     \else
7141         \def\__notesslideslabel{}
7142         \def\currentsectionlevel{\omdoc@paragraph@kw}
7143     \fi% end ifcase
7144     \__notesslideslabel%\sref@label@id\__notesslideslabel
7145     \quad #2%
7146 }%
7147 \vfill%
7148 \end{frame}%
7149 }
7150 \str_if_empty:NF \l__document_structure_sfragment_id_str {
7151     \stex_ref_new_doc_target:n\l__document_structure_sfragment_id_str
7152 }
7153 }{}
7154 }

```

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

```

7155 \def\inserttheorembodyfont{\normalfont}
7156 %\bool_if:NF \c__notesslides_notes_bool {
7157 % \defbeamertemplate{theorem begin}{miko}
7158 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7159 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7160 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7161 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7162 % \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.

```

7163 % \expandafter\def\csname Parent2\endcsname{}

```

```

7164 %}
7165
7166 \AddToHook{begindocument}{% this does not work for some reasons
7167   \setbeamertemplate{theorems}[ams style]
7168 }
7169 \bool_if:NT \c__notesslides_notes_bool {
7170   \renewenvironment{columns}[1][ ]{%
7171     \par\noindent%
7172     \begin{minipage}%
7173       \slidewidth\centering\leavevmode%
7174   }{%
7175     \end{minipage}\par\noindent%
7176   }%
7177   \newsavebox\columnbox%
7178   \renewenvironment<>{column}[2][ ]{%
7179     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7180   }{%
7181     \end{minipage}\end{lrbox}\usebox\columnbox%
7182   }%
7183 }
7184 \bool_if:NTF \c__notesslides_noproblems_bool {
7185   \newenvironment{problems}{}{}
7186 }{
7187   \excludcomment{problems}
7188 }

```

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.

```

7189 \gdef\printexcursions{}
7190 \newcommand\excursionref[2]{% label, text
7191   \bool_if:NT \c__notesslides_notes_bool {
7192     \begin{sparagraph}[title=Excursion]
7193       #2 \sref[fallback=the appendix]{#1}.
7194     \end{sparagraph}
7195   }
7196 }
7197 \newcommand\activate@excursion[2][ ]{
7198   \gappto\printexcursions{\inputref[#1]{#2}}
7199 }
7200 \newcommand\excursion[4][ ]{% repos, label, path, text
7201   \bool_if:NT \c__notesslides_notes_bool {
7202     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7203   }
7204 }

```

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

\excursiongroup

```

7205 \keys_define:nn{notesslides / excursiongroup }{

```

```

7206 id .str_set_x:N = \l_notesslides_excursion_id_str,
7207 intro .tl_set:N = \l_notesslides_excursion_intro_tl,
7208 mhrepos .str_set_x:N = \l_notesslides_excursion_mhrepos_str
7209 }
7210 \cs_new_protected:Nn \l_notesslides_excursion_args:n {
7211 \tl_clear:N \l_notesslides_excursion_intro_tl
7212 \str_clear:N \l_notesslides_excursion_id_str
7213 \str_clear:N \l_notesslides_excursion_mhrepos_str
7214 \keys_set:nn {notesslides / excursiongroup} { #1 }
7215 }
7216 \newcommand\excursiongroup[1][ ]{
7217 \l_notesslides_excursion_args:n{ #1 }
7218 \ifdefempty\printexcursions{}% only if there are excursions
7219 {\begin{note}
7220 \begin{sfragment}[#1]{Excursions}%
7221 \ifdefempty\l_notesslides_excursion_intro_tl}{
7222 \inputref[\l_notesslides_excursion_mhrepos_str]{
7223 \l_notesslides_excursion_intro_tl
7224 }
7225 }
7226 \printexcursions%
7227 \end{sfragment}
7228 \end{note}}
7229 }
7230 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{}\fi
7231 \end{package}

```

(End definition for \excursiongroup. This function is documented on page 56.)

Chapter 39

The Implementation

39.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
7232 <*package>
7233 <@@=problems>
7234 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7235 \RequirePackage{13keys2e,stex}
7236
7237 \keys_define:nn { problem / pkg }{
7238   notes      .default:n    = { true },
7239   notes      .bool_set:N   = \c__problems_notes_bool,
7240   gnotes     .default:n    = { true },
7241   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7242   hints      .default:n    = { true },
7243   hints      .bool_set:N   = \c__problems_hints_bool,
7244   solutions  .default:n    = { true },
7245   solutions  .bool_set:N   = \c__problems_solutions_bool,
7246   pts        .default:n    = { true },
7247   pts        .bool_set:N   = \c__problems_pts_bool,
7248   min        .default:n    = { true },
7249   min        .bool_set:N   = \c__problems_min_bool,
7250   boxed      .default:n    = { true },
7251   boxed      .bool_set:N   = \c__problems_boxed_bool,
7252   unknown    .code:n       = {}
7253 }
7254 \newif\ifsolutions
7255
7256 \ProcessKeysOptions{ problem / pkg }
7257 \bool_if:NTF \c__problems_solutions_bool {
7258   \solutionstrue
7259 }{
7260   \solutionsfalse
7261 }
```

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

```
7262 \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.

```
7263 \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.

```
7264 \def\prob@problem@kw{Problem}
7265 \def\prob@solution@kw{Solution}
7266 \def\prob@hint@kw{Hint}
7267 \def\prob@note@kw{Note}
7268 \def\prob@gnote@kw{Grading}
7269 \def\prob@pt@kw{pt}
7270 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7271 \AddToHook{begindocument}{
7272   \ltx@ifpackageloaded{babel}{
7273     \makeatletter
7274     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7275     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7276       \input{problem-ngerman.ldf}
7277     }
7278     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7279       \input{problem-finnish.ldf}
7280     }
7281     \clist_if_in:NnT \l_tmpa_clist {french}{
7282       \input{problem-french.ldf}
7283     }
7284     \clist_if_in:NnT \l_tmpa_clist {russian}{
7285       \input{problem-russian.ldf}
7286     }
7287     \makeatother
7288   }{}
7289 }
```

39.2 Problems and Solutions

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

```
7290 \keys_define:nn{ problem / problem }{
7291   id      .str_set_x:N = \l__problems_prob_id_str,
7292   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7293   min     .tl_set:N    = \l__problems_prob_min_tl,
7294   title   .tl_set:N    = \l__problems_prob_title_tl,
7295   type    .tl_set:N    = \l__problems_prob_type_tl,
7296   imports .tl_set:N    = \l__problems_prob_imports_tl,
7297   name    .str_set_x:N = \l__problems_prob_name_str,
7298   refnum  .int_set:N   = \l__problems_prob_refnum_int
```

```

7299 }
7300 \cs_new_protected:Nn \__problems_prob_args:n {
7301   \str_clear:N \l__problems_prob_id_str
7302   \str_clear:N \l__problems_prob_name_str
7303   \tl_clear:N \l__problems_prob_pts_tl
7304   \tl_clear:N \l__problems_prob_min_tl
7305   \tl_clear:N \l__problems_prob_title_tl
7306   \tl_clear:N \l__problems_prob_type_tl
7307   \tl_clear:N \l__problems_prob_imports_tl
7308   \int_zero_new:N \l__problems_prob_refnum_int
7309   \keys_set:nn { problem / problem }{ #1 }
7310   \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7311     \let\l__problems_prob_refnum_int\undefined
7312   }
7313 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7314 \newcounter{problem}[section]
7315 \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.

```

7316 \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

```

7317 \newcommand\prob@number{
7318   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7319     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7320   }{
7321     \int_if_exist:NTF \l__problems_prob_refnum_int {
7322       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7323     }{
7324       \prob@label\theproblem
7325     }
7326   }
7327 }

```

(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.

```

7328 \newcommand\prob@title[3]{%
7329   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7330     #2 \l__problems_inclprob_title_tl #3
7331   }{
7332     \tl_if_exist:NTF \l__problems_prob_title_tl {
7333       #2 \l__problems_prob_title_tl #3
7334     }{
7335       #1

```



```

7336     }
7337   }
7338 }

```

(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.

```

7339 \def\prob@heading{
7340   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7341   %\sref@label@id{\prob@problem@kw~\prob@number}{~}
7342 }

```

(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`

```

7343 \newenvironment{sproblem}[1][{}]{
7344   \__problems_prob_args:n{#1}%\sref@target%
7345   \@in@omtexttrue% we are in a statement (for inline definitions)
7346   \stepcounter{problem}\record@problem
7347   \def\current@section@level{\prob@problem@kw}
7348
7349   \str_if_empty:NT \l__problems_prob_name_str {
7350     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
7351     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
7352     \seq_get_left:NN \l_tmpa_seq \l__problems_prob_name_str
7353   }
7354   \exp_args:Nno\stex_module_setup:nn{type=problem}\l__problems_prob_name_str
7355
7356   \stex_reactivate_macro:N \STEXexport
7357   \stex_reactivate_macro:N \importmodule
7358   \stex_reactivate_macro:N \symdecl
7359   \stex_reactivate_macro:N \notation
7360   \stex_reactivate_macro:N \symdef
7361
7362   \stex_if_do_html:T{
7363     \begin{stex_annotate_env} {problem} {
7364       \l_stex_module_ns_str ? \l_stex_module_name_str
7365     }
7366
7367     \stex_annotate_invisible:nnn{header}{~} {
7368       \stex_annotate:nnn{language}{~} \l_stex_module_lang_str }{~}
7369       \stex_annotate:nnn{signature}{~} \l_stex_module_sig_str }{~}
7370       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
7371         \stex_annotate:nnn{metatheory}{~} \l_stex_module_meta_str }{~}
7372     }
7373   }
7374 }
7375

```

```

7376 \stex_csl_to_imports:No \importmodule \l__problems_prob_imports_tl
7377
7378
7379 \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7380   \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7381 }{
7382   \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7383 }
7384 \str_if_exist:NTF \l__problems_inclprob_id_str {
7385   \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7386 }{
7387   \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7388 }
7389
7390
7391 \stex_if_smsmode:F {
7392   \clist_set:No \l_tmpa_clist \sproblemtype
7393   \tl_clear:N \l_tmpa_tl
7394   \clist_map_inline:Nn \l_tmpa_clist {
7395     \tl_if_exist:cT {__problems_sproblem_##1_start:}{
7396       \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_start:}}
7397     }
7398   }
7399   \tl_if_empty:NTF \l_tmpa_tl {
7400     \__problems_sproblem_start:
7401   }{
7402     \l_tmpa_tl
7403   }
7404 }
7405 \stex_ref_new_doc_target:n \sproblemid
7406 \stex_smsmode_do:
7407 }{
7408   \__stex_modules_end_module:
7409   \stex_if_smsmode:F{
7410     \clist_set:No \l_tmpa_clist \sproblemtype
7411     \tl_clear:N \l_tmpa_tl
7412     \clist_map_inline:Nn \l_tmpa_clist {
7413       \tl_if_exist:cT {__problems_sproblem_##1_end:}{
7414         \tl_set:Nn \l_tmpa_tl {\use:c{__problems_sproblem_##1_end:}}
7415       }
7416     }
7417     \tl_if_empty:NTF \l_tmpa_tl {
7418       \__problems_sproblem_end:
7419     }{
7420       \l_tmpa_tl
7421     }
7422   }
7423   \stex_if_do_html:T{
7424     \end{stex_annotate_env}
7425   }
7426
7427 \smallskip
7428 }
7429

```

```

7430 \seq_put_right:Nx\g_stex_smsmode_allowedenvs_seq{\tl_to_str:n{sproblem}}
7431
7432
7433
7434 \cs_new_protected:Nn \__problems_sproblem_start: {
7435   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7436 }
7437 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7438
7439 \newcommand\stexpatchproblem[3][] {
7440   \str_set:Nx \l_tmpa_str{ #1 }
7441   \str_if_empty:NTF \l_tmpa_str {
7442     \tl_set:Nn \__problems_sproblem_start: { #2 }
7443     \tl_set:Nn \__problems_sproblem_end: { #3 }
7444   }{
7445     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7446     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7447   }
7448 }
7449
7450
7451 \bool_if:NT \c__problems_boxed_bool {
7452   \surroundwithhmdframed{problem}
7453 }

```

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

```

7454 \def\record@problem{
7455   \protected@write\@auxout{}
7456   {
7457     \string\@problem{\prob@number}
7458     {
7459       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7460         \l__problems_inclprob_pts_tl
7461       }{
7462         \l__problems_prob_pts_tl
7463       }
7464     }%
7465     {
7466       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7467         \l__problems_inclprob_min_tl
7468       }{
7469         \l__problems_prob_min_tl
7470       }
7471     }
7472   }
7473 }

```

(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).

```

7474 \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.

```

7475 \keys_define:nn { problem / solution }{
7476   id          .str_set_x:N = \l__problems_solution_id_str ,
7477   for         .tl_set:N    = \l__problems_solution_for_tl ,
7478   height      .dim_set:N   = \l__problems_solution_height_dim ,
7479   creators    .clist_set:N = \l__problems_solution_creators_clist ,
7480   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
7481   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
7482 }
7483 \cs_new_protected:Nn \__problems_solution_args:n {
7484   \str_clear:N \l__problems_solution_id_str
7485   \tl_clear:N \l__problems_solution_for_tl
7486   \tl_clear:N \l__problems_solution_srccite_tl
7487   \clist_clear:N \l__problems_solution_creators_clist
7488   \clist_clear:N \l__problems_solution_contributors_clist
7489   \dim_zero:N \l__problems_solution_height_dim
7490   \keys_set:nn { problem / solution }{ #1 }
7491 }

```

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

```

7492 \newcommand\@startsolution[1][ ]{
7493   \__problems_solution_args:n { #1 }
7494   \@in@omtexttrue% we are in a statement.
7495   \bool_if:NF \c__problems_boxed_bool { \hrule }
7496   \smallskip\noindent
7497   {\textbf\prob@solution@kw : \enspace}
7498   \begin{small}
7499   \def\current@section@level{\prob@solution@kw}
7500   \ignorespacesandpars
7501 }

```

\startolutions for the `\startolutions` 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.

```

7502 \box_new:N \l__problems_solution_box
7503 \newenvironment{solution}[1][ ]{
7504   \stex_html_backend:TF{
7505     \stex_if_do_html:T{
7506       \begin{stex_annotate_env}{solution}{ }
7507     }
7508   }{
7509     \setbox\l__problems_solution_box\vbox\bgroup
7510     \par\smallskip\hrule\smallskip
7511     \noindent\textbf{Solution:}~
7512   }
7513 }{
7514   \stex_html_backend:TF{
7515     \stex_if_do_html:T{
7516       \end{stex_annotate_env}
7517     }
7518   }{

```

```

7519 \smallskip\hrule
7520 \egroup
7521 \bool_if:NT \c__problems_solutions_bool {
7522   \box\l__problems_solution_box
7523 }
7524 }
7525 }
7526
7527 \newcommand\startsolutions{
7528   \bool_set_true:N \c__problems_solutions_bool
7529   % \specialcomment{solution}{\@startsolution}{
7530   %   \bool_if:NF \c__problems_boxed_bool {
7531   %     \hrule\medskip
7532   %   }
7533   %   \end{small}%
7534   % }
7535   % \bool_if:NT \c__problems_boxed_bool {
7536   %   \surroundwithmdframed{solution}
7537   % }
7538 }

```

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

\stopsolutions

```

7539 \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.

```

7540 \ifsolutions
7541   \startsolutions
7542 \else
7543   \stopsolutions
7544 \fi

```

exnote

```

7545 \bool_if:NTF \c__problems_notes_bool {
7546   \newenvironment{exnote}[1][]{
7547     \par\smallskip\hrule\smallskip
7548     \noindent\textbf{\prob@note@kw :~ }\small
7549   }{
7550     \smallskip\hrule
7551   }
7552 }{
7553   \excludecomment{exnote}
7554 }

```

hint

```

7555 \bool_if:NTF \c__problems_notes_bool {
7556   \newenvironment{hint}[1][]{
7557     \par\smallskip\hrule\smallskip
7558     \noindent\textbf{\prob@hint@kw :~ }\small
7559   }{
7560     \smallskip\hrule
7561   }

```

```

7562 \newenvironment{exhint}[1][]{
7563   \par\smallskip\hrule\smallskip
7564   \noindent\textbf{\prob@hint@kw :~ }\small
7565 }{
7566   \smallskip\hrule
7567 }
7568 }{
7569   \excludecomment{hint}
7570   \excludecomment{exhint}
7571 }

```

gnote

```

7572 \bool_if:NTF \c__problems_notes_bool {
7573   \newenvironment{gnote}[1][]{
7574     \par\smallskip\hrule\smallskip
7575     \noindent\textbf{\prob@gnote@kw :~ }\small
7576 }{
7577   \smallskip\hrule
7578 }
7579 }{
7580   \excludecomment{gnote}
7581 }

```

39.3 Multiple Choice Blocks

EdN:21

mcb 21

```

7582 \newenvironment{mcb}{
7583   \begin{enumerate}
7584 }{
7585   \end{enumerate}
7586 }

```

we define the keys for the mcb macro

```

7587 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7588   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7589     \bool_set_true:N #1
7590   }{
7591     \bool_set_false:N #1
7592   }
7593 }
7594 \keys_define:nn { problem / mcb }{
7595   id .str_set:N = \l__problems_mcc_id_str ,
7596   feedback .tl_set:N = \l__problems_mcc_feedback_tl ,
7597   T .default:n = { false } ,
7598   T .bool_set:N = \l__problems_mcc_t_bool ,
7599   F .default:n = { false } ,
7600   F .bool_set:N = \l__problems_mcc_f_bool ,
7601   Ttext .tl_set:N = \l__problems_mcc_Ttext_str ,
7602   Ftext .tl_set:N = \l__problems_mcc_Ftext_str
7603 }
7604 \cs_new_protected:Nn \l__problems_mcc_args:n {

```

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

```

7605 \str_clear:N \l__problems_mcc_id_str
7606 \tl_clear:N \l__problems_mcc_feedback_tl
7607 \bool_set_false:N \l__problems_mcc_t_bool
7608 \bool_set_false:N \l__problems_mcc_f_bool
7609 \tl_clear:N \l__problems_mcc_Ttext_tl
7610 \tl_clear:N \l__problems_mcc_Ftext_tl
7611 \str_clear:N \l__problems_mcc_id_str
7612 \keys_set:nn { problem / mcc }{ #1 }
7613 }

```

\mcc

```

7614 \def\mccTrueText{\textbf{(true)~}}
7615 \def\mccFalseText{\textbf{(false)~}}
7616 \newcommand\mcc[2][] {
7617   \l__problems_mcc_args:n{ #1 }
7618   \item[$\Box$] #2
7619   \ifsolutions
7620     \l
7621     \bool_if:NT \l__problems_mcc_t_bool {
7622       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccTrueText\l__problems_mcc_Ttext_tl
7623     }
7624     \bool_if:NT \l__problems_mcc_f_bool {
7625       \tl_if_empty:NTF\l__problems_mcc_Ttext_tl\mccFalseText\l__problems_mcc_Ftext_tl
7626     }
7627     \tl_if_empty:NF \l__problems_mcc_feedback_tl {
7628       \emph{(\l__problems_mcc_feedback_tl)}
7629     }
7630   \fi
7631 } %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.

```

7632
7633 \keys_define:nn{ problem / inclproblem }{
7634   id      .str_set_x:N = \l__problems_inclprob_id_str,
7635   pts     .tl_set:N    = \l__problems_inclprob_pts_tl,
7636   min     .tl_set:N    = \l__problems_inclprob_min_tl,
7637   title   .tl_set:N    = \l__problems_inclprob_title_tl,
7638   refnum  .int_set:N    = \l__problems_inclprob_refnum_int,
7639   type    .tl_set:N    = \l__problems_inclprob_type_tl,
7640   mhrepos .str_set_x:N = \l__problems_inclprob_mhrepos_str
7641 }
7642 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7643   \str_clear:N \l__problems_prob_id_str
7644   \tl_clear:N \l__problems_inclprob_pts_tl
7645   \tl_clear:N \l__problems_inclprob_min_tl
7646   \tl_clear:N \l__problems_inclprob_title_tl
7647   \tl_clear:N \l__problems_inclprob_type_tl

```

```

7648 \int_zero_new:N \l__problems_inclprob_refnum_int
7649 \str_clear:N \l__problems_inclprob_mhrepos_str
7650 \keys_set:nn { problem / inclproblem }{ #1 }
7651 \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7652   \let\l__problems_inclprob_pts_tl\undefined
7653 }
7654 \tl_if_empty:NT \l__problems_inclprob_min_tl {
7655   \let\l__problems_inclprob_min_tl\undefined
7656 }
7657 \tl_if_empty:NT \l__problems_inclprob_title_tl {
7658   \let\l__problems_inclprob_title_tl\undefined
7659 }
7660 \tl_if_empty:NT \l__problems_inclprob_type_tl {
7661   \let\l__problems_inclprob_type_tl\undefined
7662 }
7663 \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7664   \let\l__problems_inclprob_refnum_int\undefined
7665 }
7666 }
7667
7668 \cs_new_protected:Nn \__problems_inclprob_clear: {
7669   \let\l__problems_inclprob_id_str\undefined
7670   \let\l__problems_inclprob_pts_tl\undefined
7671   \let\l__problems_inclprob_min_tl\undefined
7672   \let\l__problems_inclprob_title_tl\undefined
7673   \let\l__problems_inclprob_type_tl\undefined
7674   \let\l__problems_inclprob_refnum_int\undefined
7675   \let\l__problems_inclprob_mhrepos_str\undefined
7676 }
7677 \__problems_inclprob_clear:
7678
7679 \newcommand\includeproblem[2][]{
7680   \__problems_inclprob_args:n{ #1 }
7681   \exp_args:No \stex_in_repository:nn\l__problems_inclprob_mhrepos_str{
7682     \stex_html_backend:TF {
7683       \str_clear:N \l_tmpa_str
7684       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
7685         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
7686       }
7687       \stex_annotate_invisible:nnn{includeproblem}{
7688         \l_tmpa_str / #2
7689       }{}
7690     }{
7691       \begingroup
7692       \inputreftrue
7693       \tl_if_empty:nTF{ ##1 }{
7694         \input{#2}
7695       }{
7696         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
7697       }
7698     }
7699   }
7700 }
7701 \__problems_inclprob_clear:

```



```
7702 }
```

(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.

```
7703 \AddToHook{enddocument}{
7704   \bool_if:NT \c__problems_pts_bool {
7705     \message{Total:~\arabic{pts}~points}
7706   }
7707   \bool_if:NT \c__problems_min_bool {
7708     \message{Total:~\arabic{min}~minutes}
7709   }
7710 }
```

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

```
7711 \def\pts#1{
7712   \bool_if:NT \c__problems_pts_bool {
7713     \marginpar{#1~\prob@pt@kw}
7714   }
7715 }
7716 \def\min#1{
7717   \bool_if:NT \c__problems_min_bool {
7718     \marginpar{#1~\prob@min@kw}
7719   }
7720 }
```

`\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.

```
7721 \newcounter{pts}
7722 \def\show@pts{
7723   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7724     \bool_if:NT \c__problems_pts_bool {
7725       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7726       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7727     }
7728   }{
7729     \tl_if_exist:NT \l__problems_prob_pts_tl {
7730       \bool_if:NT \c__problems_pts_bool {
7731         \tl_if_empty:NTF \l__problems_prob_pts_tl{
7732           \tl_set:Nn \l__problems_prob_pts_tl {0}
7733         }
7734         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7735         \addtocounter{pts}{\l__problems_prob_pts_tl}
7736       }
7737     }
7738   }
7739 }
```

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

\show@min

```

7740 \newcounter{min}
7741 \def\show@min{
7742   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7743     \bool_if:NT \c__problems_min_bool {
7744       \marginpar{\l__problems_inclprob_pts_tl\ min}
7745       \addtocounter{min}{\l__problems_inclprob_min_tl}
7746     }
7747   }{
7748     \tl_if_exist:NT \l__problems_prob_min_tl {
7749       \bool_if:NT \c__problems_min_bool {
7750         \tl_if_empty:NT\l__problems_prob_min_tl{
7751           \tl_set:Nn \l__problems_prob_min_tl {0}
7752         }
7753         \marginpar{\l__problems_prob_min_tl\ min}
7754         \addtocounter{min}{\l__problems_prob_min_tl}
7755       }
7756     }
7757   }
7758 }
7759 \</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.

```
7760 \*package>
7761 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7762 \RequirePackage{13keys2e}
7763
7764 \newif\iftest\testfalse
7765 \DeclareOption{test}{\testtrue}
7766 \newif\ifmultiple\multiplefalse
7767 \DeclareOption{multiple}{\multipletrue}
7768 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7769 \ProcessOptions
```

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

```
7770 \RequirePackage{keyval}[1997/11/10]
7771 \RequirePackage{problem}
```

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

```
7772 \newcommand\hwexam@assignment@kw{Assignment}
7773 \newcommand\hwexam@given@kw{Given}
7774 \newcommand\hwexam@due@kw{Due}
7775 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7776 blank~for~extra~space}
7777 \def\hwexam@minutes@kw{minutes}
7778 \newcommand\correction@probs@kw{prob.}
7779 \newcommand\correction@pts@kw{total}
7780 \newcommand\correction@reached@kw{reached}
7781 \newcommand\correction@sum@kw{Sum}
7782 \newcommand\correction@grade@kw{grade}
7783 \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

```

7784 \AddToHook{begindocument}{
7785 \ltx@ifpackageloaded{babel}{
7786 \makeatletter
7787 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7788 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7789 \input{hwexam-ngerman.ldf}
7790 }
7791 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7792 \input{hwexam-finnish.ldf}
7793 }
7794 \clist_if_in:NnT \l_tmpa_clist {french}{
7795 \input{hwexam-french.ldf}
7796 }
7797 \clist_if_in:NnT \l_tmpa_clist {russian}{
7798 \input{hwexam-russian.ldf}
7799 }
7800 \makeatother
7801 }{}
7802 }
7803

```

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.

```

7804 \newcounter{assignment}
7805 %\numberproblemsin{assignment}

We will prepare the keyval support for the assignment environment.

7806 \keys_define:nn { hwexam / assignment } {
7807 id .str_set:N = \l_@@_assign_id_str,
7808 number .int_set:N = \l_@@_assign_number_int,
7809 title .tl_set:N = \l_@@_assign_title_tl,
7810 type .tl_set:N = \l_@@_assign_type_tl,
7811 given .tl_set:N = \l_@@_assign_given_tl,
7812 due .tl_set:N = \l_@@_assign_due_tl,
7813 loadmodules .code:n = {
7814 \bool_set_true:N \l_@@_assign_loadmodules_bool
7815 }
7816 }
7817 \cs_new_protected:Nn \_@@_assignment_args:n {
7818 \str_clear:N \l_@@_assign_id_str
7819 \int_set:Nn \l_@@_assign_number_int {-1}
7820 \tl_clear:N \l_@@_assign_title_tl
7821 \tl_clear:N \l_@@_assign_type_tl
7822 \tl_clear:N \l_@@_assign_given_tl
7823 \tl_clear:N \l_@@_assign_due_tl
7824 \bool_set_false:N \l_@@_assign_loadmodules_bool
7825 \keys_set:nn { hwexam / assignment }{ #1 }
7826 }

```

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.

```

7827 \newcommand\given@due[2]{
7828 \bool_lazy_all:nF {
7829 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7830 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7831 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7832 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7833 }{ #1 }
7834
7835 \tl_if_empty:NTF \l_@@_inclasssign_given_tl {
7836 \tl_if_empty:NF \l_@@_assign_given_tl {
7837 \hwexam@given@kw\xspace\l_@@_assign_given_tl
7838 }
7839 }{
7840 \hwexam@given@kw\xspace\l_@@_inclasssign_given_tl
7841 }
7842
7843 \bool_lazy_or:nnF {
7844 \bool_lazy_and_p:nn {
7845 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7846 }{
7847 \tl_if_empty_p:V \l_@@_assign_due_tl
7848 }
7849 }{
7850 \bool_lazy_and_p:nn {
7851 \tl_if_empty_p:V \l_@@_inclasssign_due_tl
7852 }{
7853 \tl_if_empty_p:V \l_@@_assign_due_tl
7854 }
7855 }{ ,~ }
7856
7857 \tl_if_empty:NTF \l_@@_inclasssign_due_tl {
7858 \tl_if_empty:NF \l_@@_assign_due_tl {
7859 \hwexam@due@kw\xspace \l_@@_assign_due_tl
7860 }
7861 }{
7862 \hwexam@due@kw\xspace \l_@@_inclasssign_due_tl
7863 }
7864
7865 \bool_lazy_all:nF {
7866 { \tl_if_empty_p:V \l_@@_inclasssign_given_tl }
7867 { \tl_if_empty_p:V \l_@@_assign_given_tl }
7868 { \tl_if_empty_p:V \l_@@_inclasssign_due_tl }
7869 { \tl_if_empty_p:V \l_@@_assign_due_tl }
7870 }{ #2 }
7871 }

```

`\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.

```

7872 \newcommand\assignment@title[3]{
7873 \tl_if_empty:NTF \l_@@_inclasssign_title_tl {
7874 \tl_if_empty:NTF \l_@@_assign_title_tl {
7875 #1
7876 }{
7877 #2\l_@@_assign_title_tl#3
7878 }
7879 }{
7880 #2\l_@@_inclasssign_title_tl#3
7881 }
7882 }

```

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

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

```

7883 \newcommand\assignment@number{
7884 \int_compare:nNnTF \l_@@_inclasssign_number_int = {-1} {
7885 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7886 \arabic{assignment}
7887 } {
7888 \int_use:N \l_@@_assign_number_int
7889 }
7890 }{
7891 \int_use:N \l_@@_inclasssign_number_int
7892 }
7893 }

```

(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.

```

7894 \newenvironment{assignment}[1][]{
7895 \_@@_assignment_args:n { #1 }
7896 %\sref@target
7897 \int_compare:nNnTF \l_@@_assign_number_int = {-1} {
7898 \global\stepcounter{assignment}
7899 }{
7900 \global\setcounter{assignment}{\int_use:N\l_@@_assign_number_int}
7901 }
7902 \setcounter{problem}{0}
7903 \renewcommand\prob@label[1]{\assignment@number.##1}
7904 \def\current@section@level{\document@hwexamtype}
7905 %\sref@label{id}{\document@hwexamtype \thesection}
7906 \begin{@assignment}
7907 }{
7908 \end{@assignment}
7909 }

```

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

```

7910 \def\ass@title{
7911 {\protect\document@hwexamtype}\arabic{assignment}
7912 \assignment@title{\;(\;)\;}\; -- \given@due{}}
7913 }
7914 \ifmultiple
7915 \newenvironment{@assignment}{
7916 \bool_if:NTF \l_@@_assign_loadmodules_bool {
7917 \begin{sfragment}[loadmodules]{\ass@title}
7918 }{
7919 \begin{sfragment}{\ass@title}
7920 }
7921 }{
7922 \end{sfragment}
7923 }

```

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

```

7924 \else
7925 \newenvironment{@assignment}{
7926 \begin{center}\bf
7927 \Large\@title\strut\
7928 \document@hwexamtype\arabic{assignment}\assignment@title{\;(\;)\;}\;{\\}
7929 \large\given@due{--\;}\;{\;--}
7930 \end{center}
7931 }{}
7932 \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.

```

7933 \keys_define:nn { hwexam / inclassignment } {
7934 %id .str_set_x:N = \l_@@_assign_id_str,
7935 number .int_set:N = \l_@@_inclassign_number_int,
7936 title .tl_set:N = \l_@@_inclassign_title_tl,
7937 type .tl_set:N = \l_@@_inclassign_type_tl,
7938 given .tl_set:N = \l_@@_inclassign_given_tl,
7939 due .tl_set:N = \l_@@_inclassign_due_tl,
7940 mhrepos .str_set_x:N = \l_@@_inclassign_mhrepos_str
7941 }
7942 \cs_new_protected:Nn \_@@_inclassignment_args:n {
7943 \int_set:Nn \l_@@_inclassign_number_int {-1}
7944 \tl_clear:N \l_@@_inclassign_title_tl
7945 \tl_clear:N \l_@@_inclassign_type_tl
7946 \tl_clear:N \l_@@_inclassign_given_tl
7947 \tl_clear:N \l_@@_inclassign_due_tl
7948 \str_clear:N \l_@@_inclassign_mhrepos_str
7949 \keys_set:nn { hwexam / inclassignment }{ #1 }
7950 }
7951 \_@@_inclassignment_args:n {}
7952
7953 \newcommand\inputassignment[2][{}]{

```

```

7954 \_@@_inclassignment_args:n { #1 }
7955 \str_if_empty:NTF \l_@@_inclassign_mhrepos_str {
7956 \input{#2}
7957 }{
7958 \stex_in_repository:nn{\l_@@_inclassign_mhrepos_str}{
7959 \input{\mhpath{\l_@@_inclassign_mhrepos_str}{#2}}
7960 }
7961 }
7962 \_@@_inclassignment_args:n {}
7963 }
7964 \newcommand\includeassignment[2][ ]{
7965 \newpage
7966 \inputassignment[#1]{#2}
7967 }

```

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

40.4 Typesetting Exams

\quizheading

```

7968 \ExplSyntaxOff
7969 \newcommand\quizheading[1]{%
7970 \def\@tas{#1}%
7971 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7972 \ifx\@tas\@empty\else%
7973 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7974 \fi%
7975 }
7976 \ExplSyntaxOn

```

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

\testheading

```

7977
7978 \def\hwexamheader{\input{hwexam-default.header}}
7979
7980 \def\hwexamminutes{
7981 \tl_if_empty:NTF \testheading@duration {
7982 {\testheading@min}~\hwexam@minutes@kw
7983 }{
7984 \testheading@duration
7985 }
7986 }
7987
7988 \keys_define:nn { hwexam / testheading } {
7989 min .tl_set:N = \testheading@min,
7990 duration .tl_set:N = \testheading@duration,
7991 reqpts .tl_set:N = \testheading@reqpts,
7992 tools .tl_set:N = \testheading@tools
7993 }
7994 \cs_new_protected:Nn \_@@_testheading_args:n {
7995 \tl_clear:N \testheading@min
7996 \tl_clear:N \testheading@duration

```



```

7997 \tl_clear:N \testheading@reqpts
7998 \tl_clear:N \testheading@tools
7999 \keys_set:nn { hwexam / testheading }{ #1 }
8000 }
8001 \newenvironment{testheading}[1][ ]{
8002 \_@@_testheading_args:n{ #1 }
8003 \newcount\check@time\check@time=\testheading@min
8004 \advance\check@time by -\theassignment@totalmin
8005 \newif\if@bonuspoints
8006 \tl_if_empty:NTF \testheading@reqpts {
8007 \@bonuspointsfalse
8008 }{
8009 \newcount\bonus@pts
8010 \bonus@pts=\theassignment@totalpts
8011 \advance\bonus@pts by -\testheading@reqpts
8012 \edef\bonus@pts{\the\bonus@pts}
8013 \@bonuspointstrue
8014 }
8015 \edef\check@time{\the\check@time}
8016
8017 \makeatletter\hwexamheader\makeatother
8018 }{
8019 \newpage
8020 }

```

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

\testspace

```

8021 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

8022 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

8023 \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.

```

8024 <@=problems>
8025 \renewcommand\@problem[3]{
8026 \stepcounter{assignment@probs}
8027 \def\__problemspts{#2}
8028 \ifx\__problemspts\@empty\else
8029 \addtocounter{assignment@totalpts}{#2}
8030 \fi
8031 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
8032 \xdef\correction@probs{\correction@probs & #1}%
8033 \xdef\correction@pts{\correction@pts & #2}
8034 \xdef\correction@reached{\correction@reached &}

```

```

8035 }
8036 <@@=hwexam>

```

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

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

```

8037 \newcounter{assignment@probs}
8038 \newcounter{assignment@totalpts}
8039 \newcounter{assignment@totalmin}
8040 \def\correction@probs{\correction@probs@kw}
8041 \def\correction@pts{\correction@pts@kw}
8042 \def\correction@reached{\correction@reached@kw}
8043 \stepcounter{assignment@probs}
8044 \newcommand\correction@table{
8045 \resizebox{\textwidth}{!}{%
8046 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
8047 &\multicolumn{\theassignment@probs}{c|}{}%|
8048 {\footnotesize\correction@forgrading@kw} &\\ \hline
8049 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
8050 \correction@pts & \theassignment@totalpts & \\ \hline
8051 \correction@reached & & \[.7cm]\hline
8052 \end{tabular}}
8053 </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).