

The sTeX3 Package *

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

2022-03-30

Abstract

sTeX is a collection of L^AT_EX package that allow to markup documents semantically without leaving the document format, 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,
- 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 2022-03-30)

Contents

I	Manual	1
1	What is sTeX?	2
2	Quickstart	3
2.1	Setup	3
2.1.1	The sTeX IDE	3
2.1.2	Manual Setup	3
2.2	A First sTeX Document	4
2.2.1	OMDoc/xhtml Conversion	7
3	Creating sTeX Content	9
3.1	How Knowledge is Organized in sTeX	9
3.2	sTeX Archives	10
3.2.1	The Local MathHub-Directory	10
3.2.2	The Structure of sTeX Archives	10
3.2.3	MANIFEST.MF-Files	11
3.2.4	Using Files in sTeX Archives Directly	12
3.3	Module, Symbol and Notation Declarations	13
3.3.1	The smodule-Environment	13
3.3.2	Declaring New Symbols and Notations	14
	Operator Notations	18
3.3.3	Argument Types	18
	b-Type Arguments	19
	a-Type Arguments	19
	B-Type Arguments	21
3.3.4	Type and Definiens Components	21
3.3.5	Precedences and Automated Bracketing	22
3.3.6	Variables	24
3.3.7	Variable Sequences	25
3.4	Module Inheritance and Structures	27
3.4.1	Multilinguality and Translations	27
3.4.2	Simple Inheritance and Namespaces	28
3.4.3	The mathstructure Environment	29
3.4.4	The copymodule Environment	32
3.4.5	The interpretmodule Environment	33
3.5	Primitive Symbols (The sTeX Metatheory)	34
4	Using sTeX Symbols	35
4.1	\symref and its variants	35
4.2	Marking Up Text and On-the-Fly Notations	36
4.3	Referencing Symbols and Statements	38
5	sTeX Statements	39
5.1	Definitions, Theorems, Examples, Paragraphs	39
5.2	Proofs	41
6	Highlighting and Presentation Customizations	42

7	Additional Packages	44
7.1	Modular Document Structuring	44
7.2	Slides and Course Notes	44
7.3	Homework, Problems and Exams	44
II	Documentation	45
8	sTeX-Basics	46
8.1	Macros and Environments	46
8.1.1	HTML Annotations	46
8.1.2	Babel Languages	47
8.1.3	Auxiliary Methods	47
9	sTeX-MathHub	48
9.1	Macros and Environments	48
9.1.1	Files, Paths, URIs	48
9.1.2	MathHub Archives	49
9.1.3	Using Content in Archives	50
10	sTeX-References	51
10.1	Macros and Environments	51
10.1.1	Setting Reference Targets	51
10.1.2	Using References	52
11	sTeX-Modules	53
11.1	Macros and Environments	53
11.1.1	The <code>smodule</code> environment	55
12	sTeX-Module Inheritance	57
12.1	Macros and Environments	57
12.1.1	SMS Mode	57
12.1.2	Imports and Inheritance	58
13	sTeX-Symbols	60
13.1	Macros and Environments	60
14	sTeX-Terms	62
14.1	Macros and Environments	62
15	sTeX-Structural Features	64
15.1	Macros and Environments	64
15.1.1	Structures	64
16	sTeX-Statements	65
16.1	Macros and Environments	65

17	STeX-Proofs: Structural Markup for Proofs	66
17.1	Introduction	68
17.2	The User Interface	69
17.2.1	Package Options	69
17.2.2	Proofs and Proof steps	69
17.2.3	Justifications	69
17.2.4	Proof Structure	71
17.2.5	Proof End Markers	71
17.2.6	Configuration of the Presentation	71
17.3	Limitations	72
18	STeX-Metatheory	73
18.1	Symbols	73
III	Extensions	74
19	Tikzinput	75
19.1	Macros and Environments	75
20	document-structure: Semantic Markup for Open Mathematical Documents in L^AT_EX	76
20.1	Introduction	76
20.2	The User Interface	77
20.2.1	Package and Class Options	77
20.2.2	Document Structure	77
20.2.3	Ignoring Inputs	79
20.2.4	Structure Sharing	79
20.2.5	Global Variables	79
20.2.6	Colors	80
20.3	Limitations	80
21	NotesSlides – Slides and Course Notes	81
21.1	Introduction	81
21.2	The User Interface	81
21.2.1	Package Options	81
21.2.2	Notes and Slides	82
21.2.3	Header and Footer Lines of the Slides	83
21.2.4	Frame Images	83
21.2.5	Colors and Highlighting	84
21.2.6	Front Matter, Titles, etc.	84
21.2.7	Excursions	84
21.2.8	Miscellaneous	85
21.3	Limitations	85

22	problem.sty: An Infrastructure for formatting Problems	86
22.1	Introduction	86
22.2	The User Interface	86
22.2.1	Package Options	86
22.2.2	Problems and Solutions	87
22.2.3	Multiple Choice Blocks	88
22.2.4	Including Problems	88
22.2.5	Reporting Metadata	88
22.3	Limitations	88
23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	90
23.1	Introduction	91
23.2	The User Interface	91
23.2.1	Package and Class Options	91
23.2.2	Assignments	91
23.2.3	Typesetting Exams	91
23.2.4	Including Assignments	92
23.3	Limitations	92
IV	Implementation	94
24	gTeX-Basics Implementation	95
24.1	The gTeXDocument Class	95
24.2	Preliminaries	95
24.3	Messages and logging	96
24.4	HTML Annotations	97
24.5	Babel Languages	100
24.6	Auxiliary Methods	101
25	gTeX-MathHub Implementation	104
25.1	Generic Path Handling	104
25.2	PWD and kpsewhich	106
25.3	File Hooks and Tracking	107
25.4	MathHub Repositories	108
25.5	Using Content in Archives	112
26	gTeX-References Implementation	117
26.1	Document URIs and URLs	117
26.2	Setting Reference Targets	119
26.3	Using References	121
27	gTeX-Modules Implementation	124
27.1	The smodule environment	128
27.2	Invoking modules	133
28	gTeX-Module Inheritance Implementation	135
28.1	SMS Mode	135
28.2	Inheritance	139

29	STEX-Symbols Implementation	144
29.1	Symbol Declarations	144
29.2	Notations	151
29.3	Variables	161
30	STEX-Terms Implementation	168
30.1	Symbol Invocations	168
30.2	Terms	175
30.3	Notation Components	179
30.4	Variables	182
30.5	Sequences	183
31	STEX-Structural Features Implementation	185
31.1	Imports with modification	186
31.2	The feature environment	193
31.3	Structure	194
32	STEX-Statements Implementation	205
32.1	Definitions	205
32.2	Assertions	210
32.3	Examples	214
32.4	Logical Paragraphs	216
33	The Implementation	222
33.1	Package Options	222
33.2	Proofs	222
33.3	Justifications	233
34	STEX-Others Implementation	235
35	STEX-Metatheory Implementation	236
36	Tikzinput Implementation	239
37	document-structure.sty Implementation	241
37.1	The document-structure Class	241
37.2	Class Options	241
37.3	Beefing up the document environment	242
37.4	Implementation: document-structure Package	242
37.5	Package Options	242
37.6	Document Structure	244
37.7	Front and Backmatter	247
37.8	Global Variables	249

38 NotesSlides – Implementation	250
38.1 Class and Package Options	250
38.2 Notes and Slides	252
38.3 Header and Footer Lines	256
38.4 Frame Images	257
38.5 Colors and Highlighting	258
38.6 Sectioning	259
38.7 Excursions	262
39 The Implementation	263
39.1 Package Options	263
39.2 Problems and Solutions	264
39.3 Multiple Choice Blocks	270
39.4 Including Problems	271
39.5 Reporting Metadata	272
40 Implementation: The hwexam Class	274
40.1 Class Options	274
41 Implementation: The hwexam Package	276
41.1 Package Options	276
41.2 Assignments	277
41.3 Including Assignments	280
41.4 Typesetting Exams	281
41.5 Leftovers	283

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 easiyl be skipped on a first read.



Boxes like this one explain how some $\text{\texttt{S}\text{\textsf{T}}\text{\textsf{E}}\text{\textsf{X}}$ concept relates to the MMT/OMDoc system, philosophy or language.

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 to use semantic annotations in L^AT_EX documents,
- RuSTeX to convert `tex` sources to (semantically enriched) `xhtml`,
- The MMT software, that extracts semantic information from the thus generated `xhtml` and provides semantically informed added value services.

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

Foregoing on the sTeX IDE, we will need several pieces of software; namely:

- **The sTeX-Package** available [here](#).
sTeX is also available on CTAN and in TeXLive.
- 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](#)).

- **The Mmt System** available [here](#)¹. 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^ATeX 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 smglom` will download all `smglom` 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](#).

¹EdNOTE: For now, we require the sTeX-branch, requiring manually compiling the MMT sources

2.2 A First \LaTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

Feel free to 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 now:

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

smodule First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) 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` in the `smglom/calculus`-archive, and `realarith` in the `smglom/arithmetics`-archive. 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 \LaTeX symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the desired module available. Additionally, they “export” these symbols to all further modules which include the *current* module – i.e. if in some future module we would put `\importmodule{GeometricSeries}`, we would also have `\infinitesum` etc. at our disposal.

\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 `amsthm`.

The `\define{geometricSeries}` is the `\symname{?series}`

<u><code>\symname</code></u>	The <code>\symname</code> -command prints the name of a symbol, highlights it (based on customizable settings) and associates the text printed with the corresponding symbol. If you hover over the word <code>series</code> in the pdf output, you should see a tooltip showing the full URI of the symbol used.
<u><code>\symref</code></u>	The <code>\symname</code> -command is a special case of the more general <code>\symref</code> -command, which allows customizing the precise text associated with a symbol.
<u><code>\define</code></u> <u><code>\definiendum</code></u>	<p>The <code>sdefinition</code>-environment provides two additional macros, <code>\define</code> and <code>\definiendum</code> which behave similar to <code>\symname</code> and <code>\symref</code>, but explicitly mark the symbols as <i>being defined</i> in this environment, to allow for special highlighting.</p> <pre> \[\defeq{\geometricSeries}{\definiens{ \infinitesum{svar{n}}{1}{ \realdivide[frac]{1}{ \realpower{2}{svar{n}} } }} }\].\]</pre> <p>The next snippet – set in a math environment – uses several semantic macros imported from (or recursively via) <code>series</code> and <code>realarithmetics</code>, such as <code>\defeq</code>, <code>\infinitesum</code>, 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. <code>\realdivide[frac]{a}{b}</code> will use the explicit notation named <code>frac</code> of the semantic macro <code>\realdivide</code>, which yields $\frac{a}{b}$ instead of a/b.</p>
<u><code>\svar</code></u>	The <code>\svar{n}</code> command marks up the <code>n</code> as a variable with name <code>n</code> and notation <code>n</code> .
<u><code>\definiens</code></u>	The <code>sdefinition</code> -environment additionally provides the <code>\definiens</code> -command, which allows for explicitly marking up its argument as the <i>definiens</i> of the symbol currently being defined.

2.2.1 OMDoc/xhtml Conversion

So, if we run `pdflatex` on our document, then \LaTeX yields pretty colors and tooltips¹. But \LaTeX becomes a lot more powerful if we additionally convert our document to `xhtml`.

TODO VSCode Plugin

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

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

...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>
```

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

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

Remark 2.2.2:

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

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

Chapter 3

Creating sTeX Content

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

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

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

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

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

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

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

3.1 How Knowledge is Organized in sTeX

sTeX content is organized on multiple levels:

- sTeX **archives** (see [section 3.2](#)) contain individual `.tex`-files.
- These may contain sTeX **modules**, introduced via `\begin{smodule}{ModuleName}`.
- 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.
- sTeX **expressions** finally are built up from usages of semantic macros.

 M

 M

 T

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

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



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

- Symbol declarations induce OMDOC/MMT *constants*, with optional (formal) *type* and *definiens* components.
- Finally, $\text{\texttt{\textit{STeX}}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

3.2 $\text{\texttt{\textit{STeX}}}$ 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, $\text{\texttt{\textit{STeX}}}$ uses *archives* that determine the global namespaces for symbols and statements and make it possible for $\text{\texttt{\textit{STeX}}}$ to find content referenced via such URIs.

All $\text{\texttt{\textit{STeX}}}$ archives need to exist in the local MathHub-directory. $\text{\texttt{\textit{STeX}}}$ knows where this folder is via one of three means:

1. If the $\text{\texttt{\textit{STeX}}}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\text{\texttt{\textit{STeX}}}$ 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 $\text{\texttt{\textit{STeX}}}$ -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, $\text{\texttt{\textit{STeX}}}$ 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.

3.2.2 The Structure of $\text{\texttt{\textit{STeX}}}$ Archives

An $\text{\texttt{\textit{STeX}}}$ archive `group/name` needs to be 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 $\text{\texttt{\textit{STeX}}}$ system, it needs to be in `/user/foo/MathHub/smglom/calculus`.

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where $\text{\texttt{\textit{STeX}}}$ 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 this additional directory structure in the `source`-folder of an \TeX archive:

- `/source/mod/` – individual \TeX modules, containing symbol declarations, notations, and `\begin{paragraph}``[type=symdoc,for=...]` environments for “encyclopedic” 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 `\inputref`ing snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.

3.2.3 MANIFEST.MF-Files

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

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

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

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `source-base` or
 - `ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),
- `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. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

3.2.4 Using Files in \TeX Archives Directly

Several macros provided by \TeX allow for directly including files in repositories. These are:

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

$\backslash\text{inputref}$	$\backslash\text{inputref}$ [Some/Archive]{some/file} behaves like $\backslash\text{mhinput}$, but wraps the input in a $\backslash\text{begingroup} \dots \backslash\text{endgroup}$. When converting to xhtml , the file is not input at all, and instead an html-annotation is inserted that references the file. In the majority of cases $\backslash\text{inputref}$ is likely to be preferred over $\backslash\text{mhinput}$.
-----------------------------	---

$\backslash\text{ifinput}$	Both $\backslash\text{mhinput}$ and $\backslash\text{inputref}$ set $\backslash\text{ifinput}$ 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.
----------------------------	---

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

$\backslash\text{libinput}$	$\backslash\text{libinput}$ {some/file} searches for a file some/file in <ul style="list-style-type: none">• the <code>lib</code>-directory of the current archive, and• the <code>lib</code>-directory of a <code>meta-inf</code>-archive in (any of) the archive groups containing the current archive and include all found files in reverse order; e.g. $\backslash\text{libinput}\{\text{preamble}\}$ in a <code>.tex</code> -file in <code>smglom/calculus</code> will <i>first</i> input <code>../smglom/meta-inf/lib/preamble.tex</code> and then <code>../smglom/calculus/lib/preamble.tex</code> . Will throw an error if <i>no</i> candidate for some/file is found.
-----------------------------	---

$\backslash\text{libusepackage}$	$\backslash\text{libusepackage}$ [package-options]{some/file} searches for a file some/file.sty in the same way that $\backslash\text{libinput}$ does, but will call $\backslash\text{usepackage}$ [package-options]{path/to/some/file} instead of $\backslash\text{input}$. Will throw an error if not <i>exactly one</i> 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.

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 optional arguments, all of which are optional:

`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.

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

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 (=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.

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell \TeX explicitly which parts of the notation are *notation components* which *should* be highlighted. 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 \TeX 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.

`\setnotation`

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

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

Operator Notations

Once we have a semantic macro with arguments, such as `\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 `\notation` (or `\symdef`) with an *operator notation*, indicated with the optional argument `op=`. We can then invoke the operator notation using `\symbolname![notation-identifier]`. Since operator notations never take arguments, we do not need to use `\comp` in it, the whole notation is wrapped in a `\comp` automatically:

Example 8

Input:

```
1 \notation{newbinarysymbol}[ab,
2 op={\text{a:}\cdot\text{; b:}\cdot}]
3 {\comp{\text{a:}}#1\comp{\text{; b:}}#2}
4 \symname{newbinarysymbol} is also occasionally written
5 $\newbinarysymbol![ab]$
```

Output:

`newbinarysymbol` is also occasionally written `a: · ; b: ·`

\hookrightarrow `\symbolname!` is translated to OMDoc/MMT as `<OMS name="...?symbolname"/>`
 \rightarrow directly.
 \rightsquigarrow `T` \rightsquigarrow

3.3.3 Argument Types

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

b-Type Arguments

A **b-type** 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 **b-type** arguments behave exactly like **i-type** arguments within $\text{T}_{\text{E}}\text{X}$, but applications of binding operators, i.e. symbols with **b-type** 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=biil]
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.

a-Type Arguments

a-type 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. **a-type** 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 **a-type** 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 **a-type** 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 $a \neq b \wedge a \neq c \wedge a \neq d \wedge b \neq c \wedge b \neq d \wedge c \neq d$

B-Type Arguments

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

Example 12

Input:

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

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

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.

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

When $\text{\texttt{gTeX}}$ 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. $\text{\texttt{gTeX}}$ starts out with $p_d = \text{\texttt{\neginfprec}}$.
2. $\text{\texttt{gTeX}}$ encounters `\addition` with $p_{op} = 100$. Since $100 \not> \text{\texttt{\neginfprec}}$, it inserts no parentheses.
3. Next, $\text{\texttt{gTeX}}$ encounters the two arguments for `\addition`. Both have no specifically provided argument precedence, so $\text{\texttt{gTeX}}$ uses $p_d = p_{op} = 100$ for both and recurses.
4. Next, $\text{\texttt{gTeX}}$ 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 $\text{\texttt{gTeX}}$ again inserts no parentheses.
6. Since the notation of `\multiplication` has no explicitly set argument precedences, $\text{\texttt{gTeX}}$ uses the operator precedence for all arguments of `\multiplication`, hence sets $p_d = p_{op} = 50$ and recurses.
7. Next, $\text{\texttt{gTeX}}$ 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 $\text{\texttt{gTeX}}$ 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 $\text{\texttt{TeX}}$ 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 $\text{T}_\text{E}_\text{X}$ 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.

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

Output:

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

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^{\varn}}
6 ]{1,1}{\varn,\var}{\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

3.4.1 Multilinguality and Translations

If we load the \TeX document class or package with the option `lang=<lang>`, \TeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes \TeX 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 \TeX package option is set that allows for inferring a language, \TeX 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:
 $\begin{array}{ll} \text{---M---} & \text{\begin{smodule}[lang=<lang>]{Foo} generates a theory some/namespace?Foo} \\ \text{---M---} & \text{that only contains the “formal” part of the module – i.e. exactly the content} \\ \text{---T---} & \text{that is exported when using \importmodule.} \\ \text{---T---} & \text{Additionally, MMT generates a language theory some/namespace/Foo?<lang> that} \\ & \text{includes some/namespace?Foo and contains all the other document content – vari-} \\ & \text{able declarations, includes for each \usemodule, etc.} \end{array}$

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 therein available. Additionally the content 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, \TeX 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 \TeX 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 $\langle top-directory \rangle / some/path/Foo[. \langle lang \rangle] .tex$, or in $\langle top-directory \rangle / some/path[. \langle lang \rangle] .tex$ (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 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, ratherer, 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}{
2   universe = Int ,
3   op = addition ,
4   unit = zero
5 }{monoid}{\mathbb{Z}_{+,0}}
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: `mathstructure{<name>}` does in fact simply create 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 from (and correspond to) the constants in `<name>-structure`.
 $\hookrightarrow M$ $\hookrightarrow M$ $\rightsquigarrow T$ `\instantiate` appropriately generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned appropriately based on the key-value-list.

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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}{\varM{universe}}$
5 such that
6 $\varM{op}!:\mathstrut{\varM{universe},\varM{universe}}{\varM{universe}}$
7 and...
8
9 \varinstantiate{varMb}{universe = Int}{monoid}{M_2}
10
11 \noindent Let $\varMb!:=\mathstrut{\varMb{universe},\varMb{op}!,\varMb{unit}}{\varMb{universe}}$
12 a \symname{monoid} on $\mathbb{Z}$...

```

Output:

A **monoid** is a structure $M := \langle U, \circ, e \rangle$ such that $\circ : U \times U \rightarrow U$ and...
 Let $M_2 := \langle \mathbb{Z}, \circ, e \rangle$ a **monoid** on \mathbb{Z} ...

We will return to this 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 28

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 29

Input:

```

1 \begin{smodule}{ring}
2   \begin{copymodule}{group}{addition}
3     \renamedekl[name=universe]{universe}{runiverse}
4     \renamedekl[name=plus]{operation}{rplus}
5     \renamedekl[name=zero]{unit}{rzero}
6     \renamedekl[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    \renamedekl[name=times]{operation}{rtimes}
14    \renamedekl[name=one]{unit}{rone}
15  \end{copymodule}
16  \notation*{rtimes}[cdot,op=\cdot,prec=50]{#1 \comp\cdot #2}
17  \notation*{rone}[one]{\comp1}
18  Test: $\rtimes a\{rplus c\{rtimes de\}}$
19 \end{smodule}

```

Output:

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

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 30

Input:

```

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

```

Output:

3.5 Primitive Symbols (The $\text{\texttt{sTeX}}$ Metatheory)

TODO: metatheory documentation

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 31

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 32

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`, then \TeX checks all symbols currently in scope until it finds one, whose full URI ends with `string`. 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.

However, this also means that if we have symbols `foo` and e.g. `miraculous-foo`, then \TeX might resolve `\symname{foo}` to `miraculous-foo` if it finds this symbol first. It is therefore a good idea to prefix symbol names with a `?`, thus ensuring that \TeX will find the symbol `...?foo` rather than `...?miraculous-foo`.

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 33

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
 \rightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \rightsquigarrow `<OMV name="m"/>` as arguments.

\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 34

Input:

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

Output:

Addition is...

In deed, `\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)

Example 35

Input:

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

Output:

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.

The same syntax can be used in math mode, too, which allows us to spontaneously introduce new notations on the fly. We can activate it using the starred variants of semantic macros:

Example 36

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

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
- `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), `title=`, and `for=`.

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

Example 37

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`
`\definiens`
`\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.

`\definiens`[<optional symbolname>]{<code>} marks up <code> as being the explicit *definiens* of <optional symbolname> (in case `for=` has multiple symbols).

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

All four environments 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:

Example 38

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
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...

Now the **mathstructure monoid** 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}{  
2   universe = Int ,  
3   op = addition ,  
4   unit = zero  
5 }{monoid}{\mathbb{Z}_{+,0}}
```

...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

5.2 Proofs

TODO

²Of course, $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ 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**

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` defines some default customizations that can be used, but of course many existing L^AT_EX 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 L^AT_EX allows for semantic information.

Therefore we introduced the separate environments `sdefinition` etc. instead of using `definition` directly, and 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 L^AT_EX 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 that 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`-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 **sdefinitions** to use a predefined **definition**-environment, we can do

```

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 **STEX** 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 uses

```

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

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

Part II

Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

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

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

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

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

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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.

In all three 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 `{<repository-name>}` (or not, if `{<repository-name>}` is empty), and passes its ID on to `{<code>}` as `#1`. Switches back to the previous repository after executing `{<code>}`.

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<i>archive-ID</i>}{<i>filename</i>}</code>
	Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\mhinput</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Adds a <code>.bib</code> -file <i>filename</i> in archive <i>archive-ID</i> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code> <hr/>	<code>\libinput{<i>filename</i>}</code> Inputs <i>filename.tex</i> 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[<i>args</i>]{<i>filename</i>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<i>meta</i>{<i>args</i>}]<i>Arg</i>{<i>filename</i>}</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

<hr/> <hr/>	<code>\sref[<i><opt-args></i>]{<i><id></i>}</code> References the label with if <i><id></i> . Optional arguments: TODO
<hr/> <hr/>	<code>\srefsym[<i><opt-args></i>]{<i><symbol></i>}</code> Like <code>\sref</code> , but references the <i>canonical label</i> for the provided symbol. The canonical target is the last of the following occurring in the document: <ul style="list-style-type: none">• A <code>\definiendum</code> or <code>\definame</code> for <i><symbol></i>,• The <code>sassertion</code>, <code>sexample</code> or <code>sparagraph</code> with <code>for=<i><symbol></i></code> that generated <i><symbol></i> in the first place, or• A <code>\sparagraph</code> with <code>type=symdoc</code> and <code>for=<i><symbol></i></code>.
<hr/> <hr/>	<code>\srefsymuri{<i><URI></i>}{<i><text></i>}</code> A convenient short-hand for <code>\srefsym[linktext={<i><text></i>}] {<i><URI></i>}</code> , but requires the first argument to be a full URI already. Intended to be used in e.g. <code>\compemph@uri</code> , <code>\defemph@uri</code> , 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

 $\backslash\text{stex_import_module_uri:nn}$

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

 $\backslash\text{l_stex_import_name_str}$
 $\backslash\text{l_stex_import_archive_str}$
 $\backslash\text{l_stex_import_path_str}$
 $\backslash\text{l_stex_import_ns_str}$

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

<code>\symdecl</code>	<code>\symdecl{<i>macroname</i>}[<i>args</i>]</code>
-----------------------	--

Declares a new symbol with semantic macro `\macroname`. Optional arguments are:

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

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Chapter 15

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

The `sproof` package is part of the sTeX collection, a version of T_EX/L^AT_EX that allows to markup T_EX/L^AT_EX documents semantically without leaving the document format, essentially turning T_EX/L^AT_EX into a document format for mathematical knowledge management (MKM).

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

Contents

17.1 Introduction

The `sproof` (semantic proofs) package supplies macros and environment that allow to annotate the structure of mathematical proofs in \LaTeX files. This structure can be used by MKM systems for added-value services, either directly from the \LaTeX sources, or after translation. Even though it is part of the \LaTeX collection, it can be used independently, like its sister package `statements`.

\LaTeX is a version of $\text{\TeX}/\text{\LaTeX}$ that allows to markup $\text{\TeX}/\text{\LaTeX}$ documents semantically without leaving the document format, essentially turning $\text{\TeX}/\text{\LaTeX}$ into a document format for mathematical knowledge management (MKM).

```
\begin{sproof}[id=simple-proof]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcases}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[type=inline] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[type=inline]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[type=inline] We compute  $1+3=2^2=4$ .\end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n>1$ }
      \begin{spfstep}[type=assumption,id=ind-hyp]
        Now, we assume that the assertion is true for a certain  $k \geq 1$ ,
        i.e.  $\sum_{i=1}^k (2i-1) = k^2$ .
      \end{spfstep}
      \begin{sproofcomment}
        We have to show that we can derive the assertion for  $n=k+1$  from
        this assumption, i.e.  $\sum_{i=1}^{k+1} (2i-1) = (k+1)^2$ .
      \end{sproofcomment}
      \begin{spfstep}
        We obtain  $\sum_{i=1}^{k+1} (2i-1) = \sum_{i=1}^k (2i-1) + 2(k+1) - 1$ 
        \begin{justification}[method=arith:split-sum]
          by splitting the sum.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}
        Thus we have  $\sum_{i=1}^{k+1} (2i-1) = k^2 + 2k + 1$ 
        \begin{justification}[method=fertilize]
          by inductive hypothesis.
        \end{justification}
      \end{spfstep}
      \begin{spfstep}[type=conclusion]
        We can \begin{justification}[method=simplify]simplify\end{justification}
        the right-hand side to  $(k+1)^2$ , which proves the assertion.
      \end{spfstep}
    \end{spfcase}
  \end{spfcases}
  \begin{spfstep}[type=conclusion]
    We have considered all the cases, so we have proven the assertion.
  \end{spfstep}
\end{sproof}
```

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).²

²EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

17.2 The User Interface

17.2.1 Package Options

`showmeta` The `sproof` package takes a single option: `showmeta`. If this is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

17.2.2 Proofs and Proof steps

`sproof` The `proof` 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 `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise empty line. The `\spfidea` macro allows to give a one-paragraph description of the proof idea.

`spfsketch` For one-line proof sketches, we use the `\spfsketch` macro, which takes the `KeyVal` argument as `sproof` and another one: a natural language text that sketches the proof.

`spfstep` 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.

Note that both `\premise` and `\justarg` can be used with an empty second argument to mark up premises and arguments that are not explicitly mentioned in the text.

17.2.3 Justifications

`justification` This evidence is marked up with the `justification` environment in the `sproof` 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 `KeyVal` argument, which can have the `method` 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).

`\premise` The `\premise` macro allows to mark up part of the text as reference to an assertion that is used in the argumentation. In the example in Figure 1 we have used the `\premise` macro to identify the inductive hypothesis.

`\justarg` The `\justarg` macro is very similar to `\premise` 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 `\premise`. 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 `\justarg` macro.

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

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. □

Example 2: The formatted result of the proof in Figure 1

17.2.4 Proof Structure

subproof	The <code>pfcases</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
method	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.
spfcases	The <code>pfcases</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.
spfcase	The content of a <code>pfcases</code> environment are a sequence of case proofs marked up in the <code>pfcase</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>pfcase</code> environment is the same as that of a <code>proof</code> , i.e.
\spfcasesketch	<code>steps</code> , <code>proofcomments</code> , and <code>pfcases</code> environments. <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.
sproofcomment	The <code>sproofcomment</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> .

17.2.5 Proof End Markers

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:

\sproofend	The <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i>), then this can be obtained by specifying it using the
\sProofEndSymbol	<code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code>).
Some of the proof structuring macros above will insert proof end symbols for subproofs, 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 <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> .	

17.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.³. The proof step labels can be customized via the `\pstlabelstyle` macro:

Environment	configuration macro	value
<code>sproof</code>	<code>\spf@proof@kw</code>	Proof
<code>sketchproof</code>	<code>\spf@sketchproof@kw</code>	Proof Sketch

Figure 1: Configuration Hooks for Semantic Proof Markup

\pstlabelstyle	<code>\pstlabelstyle{<style>}</code> sets the style; see Figure ?? for an overview of styles. Package writers can add additional styles by adding a macro <code>\pst@make@label@<style></code> that takes
----------------	---

³EdNOTE: we might want to develop an extension `sproof-babel` in the future.

two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the \LaTeX `\@for...:=...\do{...}` macro; see Figure ?? for examples.

17.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the \TeX issue tracker at [\[sTeX\]](#).

1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)
2. currently proof steps are formatted by the \LaTeX `description` environment. We would like to configure this, e.g. to use the `inparaenum` environment for more condensed proofs. I am just not sure what the best user interface would be I can imagine redefining an internal environment `spf@proofstep@list` or adding a key `prooflistenv` to the `proof` environment that allows to specify the environment directly. Maybe we should do both.

Chapter 18

sTeX-Metatheory

The default meta theory for an sTeX module. Contains 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).

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.

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput

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

The `document-structure` package is part of the \S L^AT_EX collection, a version of $\text{\T E X}/\text{\L A T_EX}$ that allows to markup $\text{\T E X}/\text{\L A T_EX}$ documents semantically without leaving the document format, essentially turning $\text{\T E X}/\text{\L A T_EX}$ into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in \L A T_EX . This includes a simple structure sharing mechanism for \S L^AT_EX 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 L^AT_EX sources, or after translation.

20.1 Introduction

\S L^AT_EX is a version of $\text{\T E X}/\text{\L A T_EX}$ that allows to markup $\text{\T E X}/\text{\L A T_EX}$ documents semantically without leaving the document format, essentially turning $\text{\T E X}/\text{\L A T_EX}$ into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

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 L^AT_EX sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the \S L^AT_EX 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.⁴

20.2 The User Interface

The `document-structure` package generates two files: `document-structure.cls`, and `document-structure.sty`. The OMDoc class is a minimally changed variant of the standard `article` class that includes the functionality provided by `document-structure.sty`. The rest of the documentation pertains to the functionality introduced by `document-structure.sty`.

20.2.1 Package and Class Options

The `document-structure` class accept 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>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any \LaTeX packages

The `document-structure` package accepts the same except the first two.

20.2.2 Document Structure

<code>document</code>	The top-level <code>document</code> environment can be given key/value information by the
<code>\documentkeys</code>	<code>\documentkeys</code> macro in the preamble ³ . This can be used to give metadata about the
<code>id</code>	document. For the moment only the <code>id</code> key is used to give an identifier to the <code>omdoc</code>
<code>sfragment</code>	element resulting from the \LaTeX ML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the \LaTeX route, the <code>omgroup</code> environment is flexibly mapped to sectioning com-
	mands, inducing the proper sectioning level from the nesting of <code>omgroup</code> environments.
	Correspondingly, the <code>omgroup</code> environment takes an optional key/value argument for
	metadata followed by a regular argument for the (section) title of the <code>omgroup</code> . The op-
<code>id</code>	tional metadata argument has the keys <code>id</code> for an identifier, <code>creators</code> and <code>contributors</code>
<code>creators</code>	for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The
<code>contributors</code>	<code>short</code> allows to give a short title for the generated section. If the title contains semantic
<code>short</code>	macros, they need to be protected by <code>\protect</code> , and we need to give the <code>loadmodules</code>
<code>loadmodules</code>	key it needs no value. For instance we would have

```

\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderviv,loadmodules]{Introducing $\protect\bar$ Derivation

```

⁴EdNOTE: integrate with `latexml`'s `XMRef` in the Math mode.

³We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

`blindfragment`

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a `subsection*` as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` 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 `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter⁴ and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

```
\begin{document}
\begin{blindfragment}
\begin{blindfragment}
\begin{frontmatter}
\maketitle\newpage
\begin{sfragment}[display=flow]{Preface}
... <<preface>> ...
\end{sfragment}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindfragment}
... <<introductory remarks>> ...
\end{blindfragment}
\begin{sfragment}{Introduction}
... <<intro>> ...
\end{sfragment}
... <<more chapters>> ...
\bibliographystyle{alpha}\bibliography{kwarc}
\end{document}
```

Example 3: A typical Document Structure of a Book

`\skipomgroup`

The `\skipomgroup` “skips an `omgroup`”, 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 `\skipomgroup`.

`\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 `omgroup` environment, where we do not know which sectioning level we will end up.

⁴We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

20.2.3 Ignoring Inputs

`ignore` The `ignore` environment can be used for hiding text parts from the document structure.
`showignores` The body of the environment is not PDF or DVI output unless the `showignores` option is given to the `document-structure` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

editing One may want to hide unfinished or obsolete parts of a document

narrative/content markup In \LaTeX we mark up narrative-structured documents. In the generated OMDoc documents we want to be able to cache content objects that are not directly visible. For instance in the `statements` package [Koh20d] we use the `\inlinedef` macro to mark up phrase-level definitions, which verbalize more formal definitions. The latter can be hidden by an `ignore` and referenced by the `verbalizes` key in `\inlinedef`.

`\prematurestop` For prematurely stopping the formatting of a document, \LaTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `omgroup` 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.

`\afterprematurestop` `\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].

20.2.4 Structure Sharing

`\STRlabel` The `\STRlabel` macro takes two arguments: a label and the content and stores the content for later use by `\STRcopy[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets \LaTeX ML generate the correct reference.

`\STRsemantics` The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.⁵

20.2.5 Global Variables

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 \LaTeX preamble of the course notes file. `\setSGvar{\langle vname \rangle}{\langle text \rangle}` to set the global variable `\langle vname \rangle` to `\langle text \rangle` and `\useSGvar{\langle vname \rangle}` to reference it.

`\setSGvar`
`\useSGvar`
`\ifSGvar`

With `\ifSGvar` we can test for the contents of a global variable: the macro call

⁵EdNOTE: document LMID und LMXRef here if we decide to keep them.

`\ifSGvar{⟨vname⟩}{⟨val⟩}{⟨ctext⟩}` tests the content of the global variable `⟨vname⟩`, only if (after expansion) it is equal to `⟨val⟩`, the conditional text `⟨ctext⟩` is formatted.

20.2.6 Colors

For convenience, the `document-structure` package defines a couple of color macros for the `color` package: For instance `\blue` abbreviates `\textcolor{blue}`, so that `\blue{⟨something⟩}` writes `⟨something⟩` in blue. The macros `\red`, `\green`, `\cyan`, `\magenta`, `\brown`, `\yellow`, `\orange`, `\gray`, and finally `\black` are analogous.

20.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `TeX` GitHub repository [\[sTeX\]](#).

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` titles, then they sometimes are not defined by the time the heading is formatted. Need to look into how the headings are made.

Chapter 21

NotesSlides – Slides and Course Notes

We present a document class from which we can generate both course slides and course notes in a transparent way.

21.1 Introduction

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

21.2 The User Interface

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.

21.2.1 Package Options


The `notesslides` class takes a variety of class options:⁶

- | | |
|---|--|
| <code>slides</code>
<code>notes</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 21.2.2). |
|---|--|

<code>sectocframes</code>	<ul style="list-style-type: none"> If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated.
<code>showmeta</code>	<ul style="list-style-type: none"> <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 21.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>topsect</code>	<ul style="list-style-type: none"> <code>topsect=<sect></code> can be used to specify the top-level sectioning level; the default for <code><sect></code> is <code>section</code>.

21.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `notesslides` class adds the `note` environment for encapsulating the course note fragments.⁵

 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.

```
\ifnotes\maketitle\else
\frame[noframenumbering]\maketitle\fi

\begin{note}
  We start this course with ...
\end{note}

\begin{frame}
  \frametitle{The first slide}
  ...
\end{frame}
\begin{note}
  ... and more explanatory text
\end{note}

\begin{frame}
  \frametitle{The second slide}
  ...
\end{frame}
...
```

Example 4: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 4.

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between

⁶EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

⁵MK: 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.

`notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

⚠: We need to give the title frame the `noframenumbers` 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).

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 from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

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 `nomgroup`, `ndefinition`, `nexample`, `nsproof`, and `nassertion` environments.

`\inputref*`

`nparagraph`

`nfragment`
`ndefinition`
`nexample`
`nsproof`
`nassertion`

21.2.3 Header and Footer Lines of the Slides

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

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. 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.

`\setsource`

`\setlicensing`

21.2.4 Frame Images

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 \TeX notes. 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.⁷

`\frameimage`

`\mhframeimage`

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

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

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

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

⁷EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

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

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

21.2.5 Colors and Highlighting

`\textwarning` The `\textwarning` macro generates a warning sign: 

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

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:

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

```
\excursion      The \excursion{<ref>}{<path>}{<text>} is syntactic sugar for
\activateexcursion
\begin{nparagraph}[title=Excursion]
  \activateexcursion{founif}{../ex/founif}
  We will cover first-order unification in \sref{founif}.
\end{nparagraph}
```

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \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 \excursionref{<label>} for that.
```

Finally, we usually want to put the excursions into an `omgroup` environment and add an introduction, therefore we provide the a variant of the `\printexcursions` macro:

```
\excursiongroup \excursiongroup[id=<id>,intro=<path>] is equivalent to
```

```
\begin{note}
\begin{sfragment}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{sfragment}
\end{note}
```

21.2.8 Miscellaneous

21.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the \TeX GitHub repository [[sTeX](#)].

1. when option `book` which uses `\pagestyle{headings}` is given and semantic macros are given in the `omgroup` 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 `omdoc` package.

Chapter 22

problem.sty: An Infrastructure for formatting Problems

The `problem` package supplies an infrastructure that allows specify problems and to reuse them efficiently in multiple environments.

22.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁶. 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.

22.2 The User Interface

22.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> 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.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [<code>Kohlhase:mss</code>].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [<code>Kohlhase:metakeys</code>]
	for details and customization options).

⁶for the moment multiple choice problems are not supported, but may well be in a future version

22.2.2 Problems and Solutions

problem The main environment provided by the **problem** package 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. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

```
\usepackage[solutions,hints,pts,min]{problem}
\begin{document}
  \begin{sproblem}[id=elephants,pts=10,min=2,title=Fitting Elephants]
    How many Elephants can you fit into a Volkswagen beetle?
  \begin{hint}
    Think positively, this is simple!
  \end{hint}
  \begin{exnote}
    Justify your answer
  \end{exnote}
  \begin{solution}[for=elephants,height=3cm]
    Four, two in the front seats, and two in the back.
  \begin{gnote}
    if they do not give the justification deduct 5 pts
  \end{gnote}
  \end{solution}
  \end{sproblem}
\end{document}
```

Example 5: A marked up Problem

solution The **solution** environment can be to specify a solution to a problem. If the **solutions** option 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).

```
Problem 0.1 (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.


---


```

Example 6: The Formatted Problem from Figure 5

hint 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.

exnote

gnote The **gnote** (grading notes) environment can be used to document situations that

may arise in grading.

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.

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.

22.2.3 Multiple Choice Blocks

Multiple choice blocks can be formatted using the `mcb` environment, in which single choices are marked up with `\mcc[⟨keyvals⟩]{⟨text⟩}` macro, which takes an optional key/value argument `⟨keyvals⟩` for choice metadata and a required argument `⟨text⟩` for the proposed answer text. The following keys are supported

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

See Figure ?? for an example

22.2.4 Including Problems

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.

22.2.5 Reporting Metadata

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` package options are set. This allows to give students hints about the estimated time and the points to be awarded.

22.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEXGitHub` repository [[sTeX](#)].

1. none reported yet

```

\begin{sproblem}[title=Functions]
  What is the keyword to introduce a function definition in python?
  \begin{mcb}
    \mcc[T]{def}
    \mcc[F,feedback=that is for C and C++){function}
    \mcc[F,feedback=that is for Standard ML]{fun}
    \mcc[F,Ftext=Noooooooooooo,feedback=that is for Java]{public static void}
  \end{mcb}
\end{sproblem}

```

Problem 0.2 (Functions)

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

1. def
2. function
3. fun
4. public static void

Problem 0.3 (Functions)

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

1. def
!
2. function
that is for C and C++
3. fun
that is for Standard ML
4. public static void
that is for Java

Example 7: A Problem with a multiple choice block

Chapter 23

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

The `hwexam` package and class allows individual course assignment sheets and compound assignment documents using problem files marked up with the `problem` package.

Contents

23.1 Introduction

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 `problem` package [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

23.2 The User Interface

23.2.1 Package and Class Options

The `hwexam` package and class take the options `solutions`, `notes`, `hints`, `gnotes`, `pts`, `min`, and `boxed` that are just passed on to the `problems` package (cf. its documentation for a description of the intended behavior).

`showmeta` If the `showmeta` option is set, then the metadata keys are shown (see [Kohlhase:metakeys] for details and customization options).

The `hwexam` class additionally accepts the options `report`, `book`, `chapter`, `part`, and `showignores`, of the `omdoc` package [Kohlhase:smomdl] on which it is based and passes them on to that. For the `extrefs` option see [Kohlhase:sref].

23.2.2 Assignments

`assignment` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents
`number` — the ordinal of the `assignment` environment), `title` (for the assignment title; this is referenced in the title of the assignment sheet), `type` (for the assignment type; e.g. “quiz”, or “homework”), `given` (for the date the assignment was given), and `due` (for the date the assignment is due).

23.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` 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.).

`test` Finally, there is the option `test` that modifies the behavior to facilitate formatting tests. Only in `test` mode, the macros `\testspace`, `\testnewpage`, and `\testemptypage` have an effect: they generate space for the students to solve the given problems. Thus they can be left in the L^AT_EX source.

`\testspace` `\testspace` takes an argument that expands to a dimension, and leaves vertical space accordingly. `\testnewpage` makes a new page in `test` mode, and `\testemptypage` generates an empty page with the cautionary message that this page was intentionally left empty.

`testheading` Finally, the `\testheading` takes an optional keyword argument where the keys
`duration` `duration` specifies a string that specifies the duration of the test, `min` specifies the equivalent in number of minutes, and `reqpts` the points that are required for a perfect grade.
`min`
`reqpts`

23.2.4 Including Assignments

`\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.

23.3 Limitations

In this section we document known limitations. If you want to help alleviate them, please feel free to contact the package author. Some of them are currently discussed in the `STEX`GitHub repository [\[sTeX\]](#).

1. none reported yet.

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,rustex}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \rustex_if:TF {
8   \LoadClass{article}
9 }{
10   \LoadClass[border=1px,varwidth]{standalone}
11   \setlength\textwidth{15cm}
12 }
13
14 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
15 \ProcessOptions
16
17 \RequirePackage{stex}
18 </cls>
```

24.2 Preliminaries

```
19 <*package>
20
21 %%%%%%%%% basics.dtx %%%%%%%%%
22
23 \RequirePackage{expl3,l3keys2e,ltxcmds}
24 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
25
26 %\RequirePackage{morewrites}
```

```

27 %\RequirePackage{amsmath}
28
    Package options:
29 \keys_define:nn { stex } {
30   debug      .clist_set:N = \c_stex_debug_clist ,
31   lang       .clist_set:N = \c_stex_languages_clist ,
32   mathhub    .tl_set_x:N = \mathhub ,
33   sms        .bool_set:N = \c_stex_persist_mode_bool ,
34   image      .bool_set:N = \c_tikzinput_image_bool ,
35   unknown    .code:n      = {}
36 }
37 \ProcessKeysOptions { stex }

\stex The  $\TeX$  logo:
\TeX
38 \protected\def\stex{
39   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
40 }
41 \let\TeX\stex

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 46.)

24.3 Messages and logging

```

42 <@@=stex_log>

    Warnings and error messages
43 \msg_new:nnn{stex}{error/unknownlanguage}{
44   Unknown~language:~#1
45 }
46 \msg_new:nnn{stex}{warning/nomathhub}{
47   MATHHUB~system~variable~not~found~and~no~
48   \detokenize{\mathhub}~value~set!
49 }
50 \msg_new:nnn{stex}{error/deactivated-macro}{
51   The~\detokenize{#1}~command~is~only~allowed~in~#2!
52 }

\stex_debug:nn A simple macro issuing package messages with subpath.
53 \cs_new_protected:Nn \stex_debug:nn {
54   \clist_if_in:NnTF \c_stex_debug_clist { all } {
55     \msg_set:nnn{stex}{debug / #1}{
56       \\Debug~#1:~#2\\
57     }
58     \msg_none:nn{stex}{debug / #1}
59   }{
60     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
61       \msg_set:nnn{stex}{debug / #1}{
62         \\Debug~#1:~#2\\
63       }
64       \msg_none:nn{stex}{debug / #1}
65     }
66   }
67 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 46.)

Redirecting messages:

```

68 \clist_if_in:NnTF \c_stex_debug_clist {all} {
69   \msg_redirect_module:nnn{ stex }{ none }{ term }
70 }{
71   \clist_map_inline:Nn \c_stex_debug_clist {
72     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
73   }
74 }
75
76 \stex_debug:nn{log}{debug~mode~on}

```

24.4 HTML Annotations

```

77 <@=stex_annotate>
78 \RequirePackage{rustex}

```

We add the namespace abbreviation `ns:stex="http://kwarc.info/ns/sTeX"` to `RUSTEX`:

```

79 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}
80 \rustex_add_Namespace:nn{mmt}{http://uniformal.github.io/MMT}

```

Conditionals for `LATXML`:

`\if@latexml`

```

81 \ifcsname if@latexml\endcsname\else
82   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
83 \fi

```

(End definition for `\if@latexml`. This function is documented on page 46.)

`\latexml_if_p:`

`\latexml_if:TF`

```

84 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
85   \if@latexml
86     \expandafter\prg_return_true:
87   \else:
88     \expandafter\prg_return_false:
89   \fi:
90 }

```

(End definition for `\latexml_if:TF`. This function is documented on page 46.)

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

```

91 \tl_new:N \l__stex_annotate_arg_tl
92 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
93   \rustex_if:TF {
94     \rustex_direct_HTML:n { \c_ampersand_str \c_hash_str 8205; }
95   }{-}
96 }

```

(End definition for `\l__stex_annotate_arg_tl` and `\c__stex_annotate_emptyarg_tl`.)

`_stex_annotate_checkempty:n`

```

97 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
98   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
99   \tl_if_empty:NT \l__stex_annotate_arg_tl {
100     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
101   }
102 }

```

(End definition for `_stex_annotate_checkempty:n`.)

`\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 46.)

`\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 46.)

`\stex_annotate:nnx`

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).

`\stex_annotate_invisible:n`

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.

`\stex_annotate_invisible:nnn`

```

120 \rustex_if:TF{
121   \cs_new_protected:Nn \stex_annotate:nnn {
122     \_stex_annotate_checkempty:n { #3 }
123     \rustex_annotate_HTML:nn {
124       property="stex:#1" ~
125       resource="#2"
126     } {
127       \mode_if_vertical:TF{
128         \tl_use:N \l__stex_annotate_arg_tl\par
129       }{
130         \tl_use:N \l__stex_annotate_arg_tl
131       }
132     }
133   }
134   \cs_new_protected:Nn \stex_annotate_invisible:n {

```

```

135 \__stex_annotate_checkempty:n { #1 }
136 \rustex_annotate_HTML:nn {
137   stex:visible="false" ~
138   style:display="none"
139 } {
140   \mode_if_vertical:TF{
141     \tl_use:N \l__stex_annotate_arg_tl\par
142   }{
143     \tl_use:N \l__stex_annotate_arg_tl
144   }
145 }
146 }
147 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
148   \__stex_annotate_checkempty:n { #3 }
149   \rustex_annotate_HTML:nn {
150     property="stex:#1" ~
151     resource="#2" ~
152     stex:visible="false" ~
153     style:display="none"
154   } {
155     \mode_if_vertical:TF{
156       \tl_use:N \l__stex_annotate_arg_tl\par
157     }{
158       \tl_use:N \l__stex_annotate_arg_tl
159     }
160   }
161 }
162 \NewDocumentEnvironment{stex_annotate_env} { m m } {
163   \par
164   \rustex_annotate_HTML_begin:n {
165     property="stex:#1" ~
166     resource="#2"
167   }
168 }{
169   \par\rustex_annotate_HTML_end:
170 }
171 }{
172   \latexml_if:TF {
173     \cs_new_protected:Nn \stex_annotate:nnn {
174       \__stex_annotate_checkempty:n { #3 }
175       \mode_if_math:TF {
176         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
177           \tl_use:N \l__stex_annotate_arg_tl
178         }
179       }{
180         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
181           \tl_use:N \l__stex_annotate_arg_tl
182         }
183       }
184     }
185     \cs_new_protected:Nn \stex_annotate_invisible:n {
186       \__stex_annotate_checkempty:n { #1 }
187       \mode_if_math:TF {
188         \cs:w latexml@invisible@math\cs_end:{

```

```

189         \tl_use:N \l__stex_annotate_arg_tl
190     }
191 } {
192     \cs:w latexml@invisible@text\cs_end:{
193         \tl_use:N \l__stex_annotate_arg_tl
194     }
195 }
196 }
197 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
198     \__stex_annotate_checkempty:n { #3 }
199     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
200         \tl_use:N \l__stex_annotate_arg_tl
201     }
202 }
203 \NewDocumentEnvironment{stex_annotate_env} { m m } {
204     \par\begin{latexml@annotateenv}{#1}{#2}
205 }{
206     \par\end{latexml@annotateenv}
207 }
208 }{
209     \cs_new_protected:Nn \stex_annotate:nnn {#3}
210     \cs_new_protected:Nn \stex_annotate_invisible:n {}
211     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
212     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
213 }
214 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`. These functions are documented on page 47.)

24.5 Babel Languages

```

215 <@@=stex_language>

```

`\c_stex_languages_prop` We store language abbreviations in two (mutually inverse) property lists:
`\c_stex_language_abbrevs_prop`

```

216 \prop_const_from_keyval:Nn \c_stex_languages_prop {
217     en = english ,
218     de = ngerman ,
219     ar = arabic ,
220     bg = bulgarian ,
221     ru = russian ,
222     fi = finnish ,
223     ro = romanian ,
224     tr = turkish ,
225     fr = french
226 }
227
228 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
229     english = en ,
230     ngerman = de ,
231     arabic = ar ,
232     bulgarian = bg ,
233     russian = ru ,
234     finnish = fi ,

```

```

235   romanian = ro ,
236   turkish  = tr ,
237   french   = fr
238 }
239 % todo: chinese simplified (zhs)
240 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 47.)

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

```

241 \clist_if_empty:NF \c_stex_languages_clist {
242   \clist_clear:N \l_tmpa_clist
243   \clist_map_inline:Nn \c_stex_languages_clist {
244     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
245       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
246     } {
247       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
248     }
249   }
250   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
251   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
252 }
253 \AtBeginDocument{
254   \bool_lazy_any:nT {
255     {\rustex_if_p:}
256     {\latexml_if_p:}
257   } {
258     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
259     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
260     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
261     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
262     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
263       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
264       \stex_debug:nn{basics} {Language~\l_tmpa_str~
265         inferred~from~file~name}
266       \stex_annotate_invisible:nnn{language}{ \l_tmpa_str }{}
267     }
268   }
269 }
270 }

```

24.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

271 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
272   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
273   \def#1{
274     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
275   }
276 }

```

(End definition for `\stex_deactivate_macro:Nn`. This function is documented on page 47.)

\stex_reactivate_macro:N

```
277 \cs_new_protected:Nn \stex_reactivate_macro:N {  
278   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname  
279 }
```

(End definition for \stex_reactivate_macro:N. This function is documented on page 47.)

\ignorespacesandpars

```
280 \protected\def\ignorespacesandpars{  
281   \begingroup\catcode13=10\relax  
282   \@ifnextchar\par{  
283     \endgroup\expandafter\ignorespacesandpars\@gobble  
284   }{  
285     \endgroup  
286   }  
287 }  
288  
289 \cs_new:Nn \stex_copy_control_sequence:NNN {  
290   \tl_set:Nx \_tmp_args_tl {\cs_argument_spec:N #2}  
291   \tl_remove_all:Nn \_tmp_args_tl {\c_hash_str}  
292   \int_set:Nn \l_tmpa_int {\tl_count:N \_tmp_args_tl}  
293  
294   \tl_clear:N \_tmp_args_tl  
295   \int_step_inline:nn \l_tmpa_int {  
296     \tl_put_right:Nx \_tmp_args_tl {{\exp_not:n{####}\exp_not:n{##1}}}  
297   }  
298  
299   \tl_set:Nn #3 {\cs_generate_from_arg_count:NNnn #1 \cs_set:Npn}  
300   \tl_put_right:Nx #3 { {\int_use:N \l_tmpa_int}{  
301     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n  
302     \exp_after:wN\exp_after:wN\exp_after:wN {  
303       \exp_after:wN #2 \_tmp_args_tl  
304     }  
305   }}  
306 } %% TODO check if this works!  
307 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {cNN}  
308 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {NcN}  
309 \cs_generate_variant:Nn \stex_copy_control_sequence:NNN {ccN}
```

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

\MMTrule

```
310 \NewDocumentCommand \MMTrule {m m}{  
311   \seq_set_split:Nnn \l_tmpa_seq , {#2}  
312   \int_zero:N \l_tmpa_int  
313   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{  
314     $\seq_map_inline:Nn \l_tmpa_seq {  
315       \int_incr:N \l_tmpa_int  
316       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}  
317     }$  
318   }  
319 }  
320  
321 \NewDocumentCommand \MMTinclude {m}{
```

```

322 \stex_annotate_invisible:nnn{import}{#1}{  

323 }  

324 \endpackage

```

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

Chapter 25

STEX -MathHub Implementation

```
325 <*package>
326
327 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
328
329 <@@=stex_path>
330
331 Warnings and error messages
332 \msg_new:nnn{stex}{error/norepository}{
333   No~archive~#1~found~in~#2
334 }
335 \msg_new:nnn{stex}{error/notinarchive}{
336   Not~currently~in~an~archive,~but~\detokenize{#1}~
337   needs~one!
338 }
339 \msg_new:nnn{stex}{error/nofile}{
340   \detokenize{#1}~could~not~find~file~#2
341 }
342 \msg_new:nnn{stex}{error/twofiles}{
343   \detokenize{#1}~found~two~candidates~for~#2
344 }
```

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`

```
343 \cs_new_protected:Nn \stex_path_from_string:Nn {
344   \str_set:Nx \l_tmpa_str { #2 }
345   \str_if_empty:NTF \l_tmpa_str {
346     \seq_clear:N #1
347   }{
348     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
349     \sys_if_platform_windows:T{
350       \seq_clear:N \l_tmpa_tl
```

```

351 \seq_map_inline:Nn #1 {
352   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
353   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
354 }
355 \seq_set_eq:NN #1 \l_tmpa_tl
356 }
357 \stex_path_canonicalize:N #1
358 }
359 }
360

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 48.)

`\stex_path_to_string:NN`
`\stex_path_to_string:N`

```

361 \cs_new_protected:Nn \stex_path_to_string:NN {
362   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
363 }
364
365 \cs_new:Nn \stex_path_to_string:N {
366   \seq_use:Nn #1 /
367 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 48.)

`\c__stex_path_dot_str` . and .., respectively.
`\c__stex_path_up_str`

```

368 \str_const:Nn \c__stex_path_dot_str {.}
369 \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.

```

370 \cs_new_protected:Nn \stex_path_canonicalize:N {
371   \seq_if_empty:NF #1 {
372     \seq_clear:N \l_tmpa_seq
373     \seq_get_left:NN #1 \l_tmpa_tl
374     \str_if_empty:NT \l_tmpa_tl {
375       \seq_put_right:Nn \l_tmpa_seq {}
376     }
377     \seq_map_inline:Nn #1 {
378       \str_set:Nn \l_tmpa_tl { ##1 }
379       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
380         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
381           \seq_if_empty:NNTF \l_tmpa_seq {
382             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
383               \c__stex_path_up_str
384             }
385           }{
386             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
387             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
388               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
389                 \c__stex_path_up_str
390               }
391             }{

```



```

392         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
393     }
394 }
395 }{
396     \str_if_empty:NF \l_tmpa_tl {
397         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
398     }
399 }
400 }
401 }
402 \seq_gset_eq:NN #1 \l_tmpa_seq
403 }
404 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 48.)

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

```

405 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
406     \seq_if_empty:NTF #1 {
407         \prg_return_false:
408     }{
409         \seq_get_left:NN #1 \l_tmpa_tl
410         \sys_if_platform_windows:TF{
411             \str_if_in:NnTF \l_tmpa_tl {:}{
412                 \prg_return_true:
413             }{
414                 \prg_return_false:
415             }
416         }{
417             \str_if_empty:NTF \l_tmpa_tl {
418                 \prg_return_true:
419             }{
420                 \prg_return_false:
421             }
422         }
423     }
424 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 48.)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

425 \str_new:N\l_stex_kpsewhich_return_str
426 \cs_new_protected:Nn \stex_kpsewhich:n {
427     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
428     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
429     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
430 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 48.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

431 \sys_if_platform_windows:TF{
432   \begingroup\escapechar=-1\catcode'\=12
433   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
434   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
435   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
436   }}{
437   \stex_kpsewhich:n{-var-value~PWD}
438   }
439
440 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
441 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
442 \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 48.)

25.3 File Hooks and Tracking

```

443 <@@=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

```

444 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

445 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
446 \stex_path_from_string:Nn \c_stex_mainfile_seq
447   \c_stex_mainfile_str

```

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 48.)

`\g_stex_currentfile_seq`

```

448 \seq_gclear_new:N\g_stex_currentfile_seq

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 49.)

`\stex_filestack_push:n`

```

449 \cs_new_protected:Nn \stex_filestack_push:n {
450   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
451   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
452     \stex_path_from_string:Nn\g_stex_currentfile_seq{
453       \c_stex_pwd_str/#1
454     }
455   }
456   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
457   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
458 }

```



```

500 }
501 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
502 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
503 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 49.)

`_stex_mathhub_do_manifest:n` Checks whether the manifest for archive #1 already exists, and if not, finds and parses the corresponding manifest file

```

504 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
505   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
506     \str_set:Nx \l_tmpa_str { #1 }
507     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
508     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
509     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
510     \_stex_mathhub_find_manifest:N \l_tmpa_seq
511     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
512       \msg_error:nnxx{stex}{error/norepository}{#1}{
513         \stex_path_to_string:N \c_stex_mathhub_str
514       }
515     } {
516       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
517     }
518   }
519 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

520 \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`:

```

521 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
522   \seq_set_eq:NN\l_tmpa_seq #1
523   \bool_set_true:N\l_tmpa_bool
524   \bool_while_do:Nn \l_tmpa_bool {
525     \seq_if_empty:NTF \l_tmpa_seq {
526       \bool_set_false:N\l_tmpa_bool
527     } {
528       \file_if_exist:nTF{
529         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
530       } {
531         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
532         \bool_set_false:N\l_tmpa_bool
533       } {
534         \file_if_exist:nTF{
535           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
536         } {
537           \seq_put_right:Nn\l_tmpa_seq{META-INF}
538           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

539         \bool_set_false:N\l_tmpa_bool
540     }{
541         \file_if_exist:nTF{
542             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
543         }{
544             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
545             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
546             \bool_set_false:N\l_tmpa_bool
547         }{
548             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
549         }
550     }
551 }
552 }
553 }
554 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
555 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

556 \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:

```

557 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
558     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
559     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
560     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
561         \str_set:Nn \l_tmpa_str {##1}
562         \exp_args:NNoo \seq_set_split:Nnn
563             \l_tmpb_seq \c_colon_str \l_tmpa_str
564         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
565             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
566                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
567             }
568             \exp_args:No \str_case:nnTF \l_tmpa_tl {
569                 {id} {
570                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
571                     { id } \l_tmpb_tl
572                 }
573                 {narration-base} {
574                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
575                     { narr } \l_tmpb_tl
576                 }
577                 {url-base} {
578                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
579                     { docurl } \l_tmpb_tl
580                 }
581                 {source-base} {
582                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
583                     { ns } \l_tmpb_tl
584                 }

```

```

585     {ns} {
586       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
587       { ns } \l_tmpb_tl
588     }
589     {dependencies} {
590       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
591       { deps } \l_tmpb_tl
592     }
593   }{}{}
594 }{}
595 }
596 \ior_close:N \c__stex_mathhub_manifest_ior
597 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

598 \cs_new_protected:Nn \stex_set_current_repository:n {
599   \stex_require_repository:n { #1 }
600   \prop_set_eq:Nc \l_stex_current_repository_prop {
601     c_stex_mathhub_#1_manifest_prop
602   }
603 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 49.)

`\stex_require_repository:n`

```

604 \cs_new_protected:Nn \stex_require_repository:n {
605   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
606     \stex_debug:nn{mathhub}{Opening~archive:~#1}
607     \_stex_mathhub_do_manifest:n { #1 }
608   }
609 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 49.)

`\l_stex_current_repository_prop`

Current MathHub repository

```

610 %\prop_new:N \l_stex_current_repository_prop
611
612 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
613 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
614   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
615 } {
616   \_stex_mathhub_parse_manifest:n { main }
617   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
618   \l_tmpa_str
619   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
620   \c_stex_mathhub_main_manifest_prop
621   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
622   \stex_debug:nn{mathhub}{Current~repository:~
623     \prop_item:Nn \l_stex_current_repository_prop {id}
624   }
625 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 49.)

`\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.

```

626 \cs_new_protected:Nn \stex_in_repository:nn {
627   \str_set:Nx \l_tmpa_str { #1 }
628   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
629   \str_if_empty:NTF \l_tmpa_str {
630     \prop_if_exist:NTF \l_stex_current_repository_prop {
631       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
632       \exp_args:Ne \l_tmpa_cs{
633         \prop_item:Nn \l_stex_current_repository_prop { id }
634       }
635     }{
636       \l_tmpa_cs{}
637     }
638   }{
639     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
640     \stex_require_repository:n \l_tmpa_str
641     \str_set:Nx \l_tmpa_str { #1 }
642     \exp_args:Nne \use:nn {
643       \stex_set_current_repository:n \l_tmpa_str
644       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
645     }{
646       \stex_debug:nn{mathhub}{switching~back~to:~
647       \prop_if_exist:NTF \l_stex_current_repository_prop {
648         \prop_item:Nn \l_stex_current_repository_prop { id }::~
649       \meaning\l_stex_current_repository_prop
650       }{
651         no~repository
652       }
653     }
654     \prop_if_exist:NTF \l_stex_current_repository_prop {
655       \stex_set_current_repository:n {
656         \prop_item:Nn \l_stex_current_repository_prop { id }
657       }
658     }{
659       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
660     }
661   }
662 }
663 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page [49](#).)

25.5 Using Content in Archives

`\mhpath`

```

664 \def \mhpath #1 #2 {
665   \exp_args:Ne \tl_if_empty:nTF{#1}{
666     \c_stex_mathhub_str /
667     \prop_item:Nn \l_stex_current_repository_prop { id }
668     / source / #2
669   }{
670     \c_stex_mathhub_str / #1 / source / #2

```

```

671 }
672 }

```

(End definition for `\mhpath`. This function is documented on page 50.)

`\inputref`
`\mhinput`

```

673 \newif \ifinputref \inputreffalse
674
675 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
676   \stex_in_repository:nn {#1} {
677     \ifinputref
678       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
679     \else
680       \inputreftrue
681       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
682     \inputreffalse
683   \fi
684 }
685 }
686 \NewDocumentCommand \mhinput { 0{} m}{
687   \__stex_mathhub_mhinput:nn{ #1 }{ #2 }
688 }
689
690 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
691   \stex_in_repository:nn {#1} {
692     \bool_lazy_any:nTF {
693       {\rustex_if_p:}
694       {\latexml_if_p:}
695     } {
696       \str_clear:N \l_tmpa_str
697       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
698         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
699       }
700       \stex_annotate_invisible:nnn{inputref}{
701         \l_tmpa_str / #2
702       }{}
703     }{
704       \begingroup
705         \inputreftrue
706         \tl_if_empty:nTF{ ##1 }{
707           \input{#2}
708         }{
709           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
710         }
711       \endgroup
712     }
713   }
714 }
715 \NewDocumentCommand \inputref { 0{} m}{
716   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
717 }

```

(End definition for `\inputref` and `\mhinput`. These functions are documented on page 50.)

\addmhbibresource

```
718 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
719   \stex_in_repository:nn {#1} {
720     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
721   }
722 }
723 \newcommand\addmhbibresource[2][]{
724   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
725 }
```

(End definition for \addmhbibresource. This function is documented on page 50.)

\libinput

```
726 \cs_new_protected:Npn \libinput #1 {
727   \prop_if_exist:NF \l_stex_current_repository_prop {
728     \msg_error:nnn{stex}{error/notinarchive}\libinput
729   }
730   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
731     \msg_error:nnn{stex}{error/notinarchive}\libinput
732   }
733   \seq_clear:N \l__stex_mathhub_libinput_files_seq
734   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
735   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
736
737   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
738     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
739     \IfFileExists{ \l_tmpa_str }{
740       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
741     }{}
742     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
743     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
744   }
745
746   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
747   \IfFileExists{ \l_tmpa_str }{
748     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
749   }{}
750
751   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
752     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
753   }{
754     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
755       \input{ ##1 }
756     }
757   }
758 }
```

(End definition for \libinput. This function is documented on page 50.)

\libusepackage

```
759 \NewDocumentCommand \libusepackage {0{} m} {
760   \prop_if_exist:NF \l_stex_current_repository_prop {
761     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
762   }
```

```

763 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
764   \msg_error:nnn{stex}{error/notinarchive}\libusepackage
765 }
766 \seq_clear:N \l__stex_mathhub_libinput_files_seq
767 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
768 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
769
770 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
771   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
772   \IfFileExists{ \l_tmpa_str.sty }{
773     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
774   }{
775     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
776     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
777   }
778
779   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
780   \IfFileExists{ \l_tmpa_str.sty }{
781     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
782   }{
783
784     \seq_if_empty:NNTF \l__stex_mathhub_libinput_files_seq {
785       \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
786     }{
787       \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
788         \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
789           \usepackage[#1]{ ##1 }
790         }
791       }{
792         \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
793       }
794     }
795   }

```

(End definition for `\libusepackage`. This function is documented on page 50.)

`\mhgraphics`
`\cmhgraphics`

```

796
797 \AddToHook{begindocument}{
798   \ltx@ifpackageloaded{graphicx}{
799     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
800     \newcommand\mhgraphics[2][{}]{%
801       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
802       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
803       \newcommand\cmhgraphics[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
804     }{

```

(End definition for `\mhgraphics` and `\cmhgraphics`. These functions are documented on page 50.)

`\lstinputmhlisting`
`\cmlstinputmhlisting`

```

805 \ltx@ifpackageloaded{listings}{
806   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
807   \newcommand\lstinputmhlisting[2][{}]{%
808     \def\lst@mhrepos{}\setkeys{lst}{#1}%
809     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}

```

```

810     \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
811   }{}
812 }
813
814 \</package>

```

(End definition for \lstinputmhlisting and \clstinputmhlisting. These functions are documented on page 50.)

Chapter 26

STEX -References Implementation

```
815 <*package>
816
817 %%%%%%%%%%% references.dtx %%%%%%%%%%%
818
819 <@@=stex_refs>
      Warnings and error messages
820
```

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

```
821 %\iow_new:N \c__stex_refs_refs_iow
822 \AddToHook{begindocument}{
823 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
824 }
825 \AddToHook{enddocument}{
826 % \iow_close:N \c__stex_refs_refs_iow
827 }
```

`\STEXreftitle`

```
828 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
829
830 \NewDocumentCommand \STEXreftitle { m } {
831 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
832 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
833 \str_new:N \l_stex_current_docns_str
```

(End definition for `\l_stex_current_docns_str`. This variable is documented on page 51.)

`\stex_get_document_uri:`

```
834 \cs_new_protected:Nn \stex_get_document_uri: {  
835   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
836   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
837   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
838   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
839   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
840  
841   \str_clear:N \l_tmpa_str  
842   \prop_if_exist:NT \l_stex_current_repository_prop {  
843     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
844       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
845     }  
846   }  
847  
848   \str_if_empty:NTF \l_tmpa_str {  
849     \str_set:Nx \l_stex_current_docns_str {  
850       file:/\stex_path_to_string:N \l_tmpa_seq  
851     }  
852   }{  
853     \bool_set_true:N \l_tmpa_bool  
854     \bool_while_do:Nn \l_tmpa_bool {  
855       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
856       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
857         {source} { \bool_set_false:N \l_tmpa_bool }  
858       }{}{  
859         \seq_if_empty:NT \l_tmpa_seq {  
860           \bool_set_false:N \l_tmpa_bool  
861         }  
862       }  
863     }  
864  
865     \seq_if_empty:NTF \l_tmpa_seq {  
866       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
867     }{  
868       \str_set:Nx \l_stex_current_docns_str {  
869         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
870       }  
871     }  
872   }  
873 }
```

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

`\l_stex_current_docurl_str`

```
874 \str_new:N \l_stex_current_docurl_str
```

(End definition for `\l_stex_current_docurl_str`. This variable is documented on page 51.)

`\stex_get_document_url:`

```
875 \cs_new_protected:Nn \stex_get_document_url: {  
876   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
877   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
878   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

879 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
880 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
881
882 \str_clear:N \l_tmpa_str
883 \prop_if_exist:NT \l_stex_current_repository_prop {
884   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
885     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
886       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
887     }
888   }
889 }
890
891 \str_if_empty:NTF \l_tmpa_str {
892   \str_set:Nx \l_stex_current_docurl_str {
893     file:/\stex_path_to_string:N \l_tmpa_seq
894   }
895 }{
896   \bool_set_true:N \l_tmpa_bool
897   \bool_while_do:Nn \l_tmpa_bool {
898     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
899     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
900       {source} { \bool_set_false:N \l_tmpa_bool }
901     }{}{
902       \seq_if_empty:NT \l_tmpa_seq {
903         \bool_set_false:N \l_tmpa_bool
904       }
905     }
906   }
907
908   \seq_if_empty:NTF \l_tmpa_seq {
909     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
910   }{
911     \str_set:Nx \l_stex_current_docurl_str {
912       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
913     }
914   }
915 }
916 }

```

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

26.2 Setting Reference Targets

```

917 \str_const:Nn \c__stex_refs_url_str{URL}
918 \str_const:Nn \c__stex_refs_ref_str{REF}
919 \str_new:N \l__stex_refs_curr_label_str
920 % @currentlabel -> number
921 % @currentlabelname -> title
922 % @currentHref -> name.number <- id of some kind
923 % \theH# -> \arabic{section}
924 % \the# -> number
925 % \hyper@makecurrent{#}
926 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

927 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
928   \stex_get_document_uri:
929   \str_clear:N \l__stex_refs_curr_label_str
930   \str_set:Nx \l_tmpa_str { #1 }
931   \str_if_empty:NT \l_tmpa_str {
932     \int_incr:N \l__stex_refs_unnamed_counter_int
933     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
934   }
935   \str_set:Nx \l__stex_refs_curr_label_str {
936     \l_stex_current_docns_str?\l_tmpa_str
937   }
938   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
939     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
940   }
941   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
942     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
943   }
944   \stex_if_smsmode:TF {
945     \stex_get_document_url:
946     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
947     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
948   }{
949     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
950     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
951     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
952     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
953   }
954 }

```

(End definition for `\stex_ref_new_doc_target:n`. This function is documented on page 51.)

The following is used to set the necessary macros in the .aux-file.

```

955 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
956   \str_set:Nn \l_tmpa_str {#1?#2}
957   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
958   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
959     \seq_new:c {g__stex_refs_labels_#2_seq}
960   }
961   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
962     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
963   }
964 }

```

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

```

965 \AtEndDocument{
966   \def\stexauxadddocref#1 #2 {}{}
967 }

```

`\stex_ref_new_sym_target:n`

```

968 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
969   \stex_if_smsmode:TF {
970     \str_if_exist:cF{sref_sym_#1_type}{
971       \stex_get_document_url:
972       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

973     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
974   }
975 }{
976   \str_if_empty:NF \l__stex_refs_curr_label_str {
977     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
978     \immediate\write\@auxout{
979       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
980         \l__stex_refs_curr_label_str
981       }
982     }
983   }
984 }
985 }

```

(End definition for `\stex_ref_new_sym_target:n`. This function is documented on page 51.)

26.3 Using References

```

986 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

987
988 \keys_define:nn { stex / sref } {
989   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
990   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
991   pre           .tl_set:N = \l__stex_refs_pre_tl ,
992   post          .tl_set:N = \l__stex_refs_post_tl ,
993 }
994 \cs_new_protected:Nn \__stex_refs_args:n {
995   \tl_clear:N \l__stex_refs_linktext_tl
996   \tl_clear:N \l__stex_refs_fallback_tl
997   \tl_clear:N \l__stex_refs_pre_tl
998   \tl_clear:N \l__stex_refs_post_tl
999   \str_clear:N \l__stex_refs_repo_str
1000   \keys_set:nn { stex / sref } { #1 }
1001 }

```

The actual macro:

```

1002 \NewDocumentCommand \sref { 0{} m}{
1003   \__stex_refs_args:n { #1 }
1004   \str_if_empty:NTF \l__stex_refs_indocument_str {
1005     \str_set:Nx \l_tmpa_str { #2 }
1006     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
1007     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
1008       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
1009         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
1010           \str_clear:N \l_tmpa_str
1011         }
1012       }{
1013         \str_clear:N \l_tmpa_str
1014       }
1015     }{
1016       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1017       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```



```

1018 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
1019 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
1020   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
1021   \str_clear:N \l_tmpa_str
1022   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
1023     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
1024       \str_range:nnn { ##1 }{-\l_tmpa_int}{ -1 }
1025     }{
1026       \seq_map_break:n {
1027         \str_set:Nn \l_tmpa_str { ##1 }
1028       }
1029     }
1030   }
1031 }{
1032   \str_clear:N \l_tmpa_str
1033 }
1034 }
1035 \str_if_empty:NTF \l_tmpa_str {
1036   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
1037 }{
1038   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
1039     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1040       \cs_if_exist:cTF{autoref}{
1041         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1042       }{
1043         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
1044       }
1045     }{
1046       \ltx@ifpackageloaded{hyperref}{
1047         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
1048       }{
1049         \l__stex_refs_linktext_tl
1050       }
1051     }
1052   }{
1053     \ltx@ifpackageloaded{hyperref}{
1054       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
1055     }{
1056       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
1057     }
1058   }
1059 }
1060 }{
1061   % TODO
1062 }
1063 }

```

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

`\srefsym`

```

1064 \NewDocumentCommand \srefsym { 0{} m}{
1065   \stex_get_symbol:n { #2 }
1066   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1067 }

```

```

1068
1069 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1070   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1071     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1072   }{
1073     \__stex_refs_args:n { #1 }
1074     \str_if_empty:NTF \l__stex_refs_indocument_str {
1075       \tl_if_exist:cTF{sref_sym_#2 _type}{
1076         % doc uri in \l_tmpb_str
1077         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1078         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1079           % reference
1080           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1081             \cs_if_exist:cTF{autoref}{
1082               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1083             }{
1084               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1085             }
1086           }{
1087             \ltx@ifpackageloaded{hyperref}{
1088               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1089             }{
1090               \l__stex_refs_linktext_tl
1091             }
1092           }
1093         }{
1094           % URL
1095           \ltx@ifpackageloaded{hyperref}{
1096             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1097           }{
1098             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1099           }
1100         }
1101       }{
1102         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1103       }
1104     }{
1105       % TODO
1106     }
1107   }
1108 }

```

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

`\srefsymuri`

```

1109 \cs_new_protected:Npn \srefsymuri #1 #2 {
1110   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1111 }

```

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

```

1112 </package>

```

Chapter 27

STEX -Modules Implementation

```
1113 <*package>
1114
1115 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1116
1117 <@@=stex_modules>
1118
1119 Warnings and error messages
1120 \msg_new:nnn{stex}{error/unknownmodule}{
1121   No~module~#1~found
1122 }
1123 \msg_new:nnn{stex}{error/syntax}{
1124   Syntax~error:~#1
1125 }
1126 \msg_new:nnn{stex}{error/siglanguage}{
1127   Module~#1~declares~signature~#2,~but~does~not~
1128   declare~its~language
1129 }
1130 \msg_new:nnn{stex}{warning/deprecated}{
1131   #1~is~deprecated;~please~use~#2~instead!
1132 }
1133 \msg_new:nnn{stex}{error/conflictingmodules}{
1134   Conflicting~imports~for~module~#1
1135 }
```

`\l_stex_current_module_str` The current module:

```
1135 \str_new:N \l_stex_current_module_str
```

(End definition for `\l_stex_current_module_str`. This variable is documented on page 54.)

`\l_stex_all_modules_seq` Stores all available modules

```
1136 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 54.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1137 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1138   \str_if_empty:NTF \l_stex_current_module_str
1139   \prg_return_false: \prg_return_true:
1140 }

(End definition for \stex_if_in_module:TF. This function is documented on page 54.)

```

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
1141 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1142   \prop_if_exist:cTF { c_stex_module_#1_prop }
1143   \prg_return_true: \prg_return_false:
1144 }

(End definition for \stex_if_module_exists:nTF. This function is documented on page 54.)

```

```

\stex_add_to_current_module:n
\STEXexport
1145 \cs_new_protected:Nn \stex_add_to_current_module:n {
1146   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1147 }
1148 \cs_new_protected:Npn \STEXexport {
1149   \begingroup
1150   \newlinechar=-1\relax
1151   \endlinechar=-1\relax
1152   %\catcode'\ = 9\relax
1153   \expandafter\endgroup\__stex_modules_export:n
1154 }
1155 \cs_new_protected:Nn \__stex_modules_export:n {
1156   \ignorespaces #1
1157   \stex_add_to_current_module:n { \ignorespaces #1 }
1158   \stex_smsmode_do:
1159 }
1160 \stex_deactivate_macro:Nn \STEXexport {module~environments}

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented
on page 54.)

```

```

\stex_add_constant_to_current_module:n
1161 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1162   \str_set:Nx \l_tmpa_str { #1 }
1163   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1164 }

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page
54.)

```

```

\stex_add_import_to_current_module:n
1165 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1166   \str_set:Nx \l_tmpa_str { #1 }
1167   \exp_args:Nno
1168   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1169     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1170   }
1171 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 54.)

`\stex_collect_imports:n`

```

1172 \cs_new_protected:Nn \stex_collect_imports:n {
1173   \seq_clear:N \l_stex_collect_imports_seq
1174   \__stex_modules_collect_imports:n {#1}
1175 }
1176 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1177   \seq_map_inline:cn {c_stex_module_#1_imports} {
1178     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1179       \__stex_modules_collect_imports:n { ##1 }
1180     }
1181   }
1182   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1183     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1184   }
1185 }

```

(End definition for `\stex_collect_imports:n`. This function is documented on page 54.)

`\stex_do_up_to_module:n`

```

1186 \int_new:N \l__stex_modules_group_depth_int
1187 \cs_new_protected:Nn \stex_do_up_to_module:n {
1188   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1189     #1
1190   }{
1191     #1
1192     \expandafter \tl_gset:Nn
1193     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1194     \expandafter\expandafter\expandafter\endcsname
1195     \expandafter\expandafter\expandafter { \csname
1196       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1197     \aftergroup\__stex_modules_aftergroup_do:
1198   }
1199 }
1200 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1201   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1202     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1203   }}
1204   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1205     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1206     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1207   }{
1208     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1209     \aftergroup\__stex_modules_aftergroup_do:
1210   }
1211 }
1212 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1213   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1214 }

```

(End definition for `\stex_do_up_to_module:n`. This function is documented on page 54.)

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

1215

(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.

```

1216 \str_new:N \l_stex_module_ns_str
1217 \str_new:N \l_stex_module_subpath_str
1218 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1219   \seq_set_eq:NN \l_tmpa_seq #2
1220   % split off file extension
1221   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str % <- filename
1222   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1223   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str % <- filename without suffixes
1224   \seq_put_right:No \l_tmpa_seq \l_tmpb_str % <- file path including name without suffixes
1225
1226   \bool_set_true:N \l_tmpa_bool
1227   \bool_while_do:Nn \l_tmpa_bool {
1228     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1229     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1230       {source} { \bool_set_false:N \l_tmpa_bool }
1231     }{}{
1232       \seq_if_empty:NT \l_tmpa_seq {
1233         \bool_set_false:N \l_tmpa_bool
1234       }
1235     }
1236   }
1237
1238   \stex_path_to_string:NN \l_tmpa_seq \l_stex_module_subpath_str
1239   % \l_tmpa_seq <- sub-path relative to archive
1240   \str_if_empty:NTF \l_stex_module_subpath_str {
1241     \str_set:Nx \l_stex_module_ns_str {#1}
1242   }{
1243     \str_set:Nx \l_stex_module_ns_str {
1244       #1/\l_stex_module_subpath_str
1245     }
1246   }
1247 }
1248
1249 \cs_new_protected:Nn \stex_modules_current_namespace: {
1250   \str_clear:N \l_stex_module_subpath_str
1251   \prop_if_exist:NTF \l_stex_current_repository_prop {
1252     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1253     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1254   }{
1255     % split off file extension
1256     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1257     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1258     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1259     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1260     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1261     \str_set:Nx \l_stex_module_ns_str {

```

```

1262     file:/\stex_path_to_string:N \l_tmpa_seq
1263   }
1264 }
1265 }

```

(End definition for `\stex_modules_current_namespace::`. This function is documented on page 55.)

27.1 The smodule environment

smodule arguments:

```

1266 \keys_define:nn { stex / module } {
1267   title      .tl_set:N      = \smodulename ,
1268   type       .str_set_x:N   = \smodulename ,
1269   id         .str_set_x:N   = \smoduleid ,
1270   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1271   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1272   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1273   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1274   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1275   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1276   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1277   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1278 }
1279
1280 \cs_new_protected:Nn \__stex_modules_args:n {
1281   \str_clear:N \smodulename
1282   \str_clear:N \smodulename
1283   \str_clear:N \smoduleid
1284   \str_clear:N \l_stex_module_ns_str
1285   \str_clear:N \l_stex_module_deprecate_str
1286   \str_clear:N \l_stex_module_lang_str
1287   \str_clear:N \l_stex_module_sig_str
1288   \str_clear:N \l_stex_module_creators_str
1289   \str_clear:N \l_stex_module_contributors_str
1290   \str_clear:N \l_stex_module_meta_str
1291   \str_clear:N \l_stex_module_srccite_str
1292   \keys_set:nn { stex / module } { #1 }
1293 }
1294
1295 % module parameters here? In the body?
1296

```

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

```

1297 \cs_new_protected:Nn \stex_module_setup:nn {
1298   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1299   \str_set:Nx \l_stex_module_name_str { #2 }
1300   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1301 \stex_if_in_module:TF {
1302   % Nested module
1303   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}

```

```

1304     { ns } \l_stex_module_ns_str
1305     \str_set:Nx \l_stex_module_name_str {
1306       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1307       { name } / \l_stex_module_name_str
1308     }
1309   }{
1310     % not nested:
1311     \str_if_empty:NT \l_stex_module_ns_str {
1312       \stex_modules_current_namespace:
1313       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1314         / {\l_stex_module_ns_str}
1315       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1316       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1317         \str_set:Nx \l_stex_module_ns_str {
1318           \stex_path_to_string:N \l_tmpa_seq
1319         }
1320       }
1321     }
1322   }

```

Next, we determine the language of the module:

```

1323   \str_if_empty:NT \l_stex_module_lang_str {
1324     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1325     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1326     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1327     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1328     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1329       \seq_pop_left: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 55.)

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_smsmode:F{
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 \smoduletype {
1455       \stex_annotate:nnn{type}{\smoduletype}{ }
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 }

```

(End definition for `_stex_modules_end_module:`)

The core environment

```

1466 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1467 \NewDocumentEnvironment { smodule } { 0 } { m } {
1468   \stex_module_setup:nn{#1}{#2}
1469   \par
1470   \stex_if_smsmode:F{
1471     \tl_clear:N \l_tmpa_tl
1472     \clist_map_inline:Nn \smoduletype {
1473       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1474         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1475       }
1476     }
1477     \tl_if_empty:NTF \l_tmpa_tl {
1478       \_stex_modules_smodule_start:
1479     }{
1480       \l_tmpa_tl
1481     }
1482   }
1483   \_stex_modules_begin_module:
1484   \str_if_empty:NF \smoduleid {
1485     \stex_ref_new_doc_target:n \smoduleid
1486   }
1487   \stex_smsmode_do:
1488 } {
1489   \_stex_modules_end_module:
1490   \stex_if_smsmode:F {
1491     \end{stex_annotate_env}
1492     \clist_set:No \l_tmpa_clist \smoduletype
1493     \tl_clear:N \l_tmpa_tl
1494     \clist_map_inline:Nn \l_tmpa_clist {
1495       \tl_if_exist:cT {\_stex_modules_smodule_##1_end:}{
1496         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_end:}}
1497       }
1498     }
1499     \tl_if_empty:NTF \l_tmpa_tl {

```

```

1500     \__stex_modules_smodule_end:
1501   }{
1502     \l_tmpa_tl
1503   }
1504 }
1505 }

```

`\stexpatchmodule`

```

1506 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1507 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1508
1509 \newcommand\stexpatchmodule[3] [] {
1510   \str_set:Nx \l_tmpa_str{ #1 }
1511   \str_if_empty:NTF \l_tmpa_str {
1512     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1513     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1514   }{
1515     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1516     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1517   }
1518 }

```

(End definition for `\stexpatchmodule`. This function is documented on page 55.)

27.2 Invoking modules

`\STEXModule` `\stex_invoke_module:n`

```

1519 \NewDocumentCommand \STEXModule { m } {
1520   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1521   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1522   \tl_set:Nn \l_tmpa_tl {
1523     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1524   }
1525   \seq_map_inline:Nn \l_stex_all_modules_seq {
1526     \str_set:Nn \l_tmpb_str { ##1 }
1527     \str_if_eq:eeT { \l_tmpa_str } {
1528       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1529     } {
1530       \seq_map_break:n {
1531         \tl_set:Nn \l_tmpa_tl {
1532           \stex_invoke_module:n { ##1 }
1533         }
1534       }
1535     }
1536   }
1537   \l_tmpa_tl
1538 }
1539
1540 \cs_new_protected:Nn \stex_invoke_module:n {
1541   \stex_debug:nn{modules}{Invoking~module~#1}
1542   \peek_charcode_remove:NTF ! {
1543     \__stex_modules_invoke_uri:nN { #1 }
1544   } {

```

```

1545 \peek_charcode_remove:NTF ? {
1546   \__stex_modules_invoke_symbol:nn { #1 }
1547 } {
1548   \msg_error:nnx{stex}{error/syntax}{
1549     ?~or~!~expected~after~
1550     \c_backslash_str STEXModule{#1}
1551   }
1552 }
1553 }
1554 }
1555
1556 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1557   \str_set:Nn #2 { #1 }
1558 }
1559
1560 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1561   \stex_invoke_symbol:n{#1?#2}
1562 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 55.)

`\stex_activate_module:n`

```

1563 \bool_new:N \l_stex_in_meta_bool
1564 \bool_set_false:N \l_stex_in_meta_bool
1565 \cs_new_protected:Nn \stex_activate_module:n {
1566   \stex_debug:nn{modules}{Activating~module~#1}
1567   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1568     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1569   }
1570   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1571     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1572     \use:c{ c_stex_module_#1_code }
1573   }
1574 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 56.)

```

1575 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1576 <*package>
1577
1578 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1579
```

28.1 SMS Mode

```
1580 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1581 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1582 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1583 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1584
1585 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1586   \makeatletter
1587   \makeatother
1588   \ExplSyntaxOn
1589   \ExplSyntaxOff
1590   \rustexBREAK
1591 }
1592
1593 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1594   \symdef
1595   \importmodule
1596   \notation
1597   \symdecl
1598   \STEXexport
1599   \inlineass
1600   \inlinedef
1601   \inlineex
1602   \endinput
1603   \setnotation
```

```

1604 \copynotation
1605 \assign
1606 \renamedekl
1607 \donotcopy
1608 \instantiate
1609 }
1610
1611 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1612   \tl_to_str:n {
1613     smodule,
1614     copymodule,
1615     interpretmodule,
1616     sdefinition,
1617     sexample,
1618     sassertion,
1619     sparagraph,
1620     mathstructure
1621   }
1622 }

```

(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 57.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF
1623 \bool_new:N \g__stex_smsmode_bool
1624 \bool_set_false:N \g__stex_smsmode_bool
1625 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1626   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1627 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 57.)

```

\_stex_smsmode_in_smsmode:nn
1628 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1629   \vbox_set:Nn \l_tmpa_box {
1630     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1631     \bool_gset_true:N \g__stex_smsmode_bool
1632     #2
1633     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1634   }
1635   \box_clear:N \l_tmpa_box
1636 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

```

\stex_file_in_smsmode:nn
1637 \quark_new:N \q__stex_smsmode_break
1638
1639 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1640   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1641   \stex_smsmode_do:
1642 }
1643
1644 \cs_new_protected:Nn \_stex_smsmode_module:nn {
1645   \_stex_modules_args:n{#1}

```

```

1646 \stex_if_in_module:F {
1647   \str_if_empty:NF \l_stex_module_sig_str {
1648     \stex_modules_current_namespace:
1649     \str_set:Nx \l_stex_module_name_str { #2 }
1650     \stex_if_module_exists:nF{\l_stex_module_ns_str?\l_stex_module_name_str}{
1651       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1652       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1653       \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1654       \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1655       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1656       \str_set:Nx \l_tmpa_str {
1657         \stex_path_to_string:N \l_tmpa_seq /
1658         \l_tmpa_str . \l_stex_module_sig_str .tex
1659       }
1660       \IfFileExists \l_tmpa_str {
1661         \exp_args:NNx \seq_gput_right:Nn \l__stex_smsmode_sigmodules_seq \l_tmpa_str
1662       }{
1663         \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1664       }
1665     }
1666   }
1667 }
1668 }
1669
1670 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1671   \stex_filestack_push:n{#1}
1672   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1673   \seq_gclear:N \l__stex_smsmode_sigmodules_seq
1674   % ----- new -----
1675   \__stex_smsmode_in_smsmode:nn{#1}{
1676     \let\importmodule\__stex_smsmode_importmodule:
1677     \let\stex_module_setup:nn\__stex_smsmode_module:nn
1678     \let\__stex_modules_begin_module:\relax
1679     \let\__stex_modules_end_module:\relax
1680     \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1681     \exp_args:NNx \seq_put_right:Nn \g_stex_smsmode_allowedenvs_seq {\tl_to_str:n{module}}
1682     \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1683     \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1684     \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1685     \everyeof{\q__stex_smsmode_break\noexpand}
1686     \expandafter\expandafter\expandafter
1687     \stex_smsmode_do:
1688     \csname @ @ input\endcsname "#1"\relax
1689
1690     \seq_map_inline:Nn \l__stex_smsmode_sigmodules_seq {
1691       \stex_filestack_push:n{##1}
1692       \expandafter\expandafter\expandafter
1693       \stex_smsmode_do:
1694       \csname @ @ input\endcsname "##1"\relax
1695       \stex_filestack_pop:
1696     }
1697   }
1698   % ----- new -----
1699   \__stex_smsmode_in_smsmode:nn{#1} {

```



```

1700 #2
1701 % ----- new -----
1702 \begingroup
1703 %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1704 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1705   \stex_import_module_uri:nn ##1
1706   \stex_import_require_module:nnnn
1707   \l_stex_import_ns_str
1708   \l_stex_import_archive_str
1709   \l_stex_import_path_str
1710   \l_stex_import_name_str
1711 }
1712 \endgroup
1713 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1714 % ----- new -----
1715 \everyeof{\q__stex_smsmode_break\noexpand}
1716 \expandafter\expandafter\expandafter
1717 \stex_smsmode_do:
1718 \csname @ @ input\endcsname "#1"\relax
1719 }
1720 \stex_filestack_pop:
1721 }

```

(End definition for `\stex_file_in_smsmode:nn`. This function is documented on page 58.)

`\stex_smsmode_do:` is executed on encountering `\` in `smsmode`. It checks whether the corresponding command is allowed and executes or ignores it accordingly:

```

1722 \cs_new_protected:Npn \stex_smsmode_do: {
1723   \stex_if_smsmode:T {
1724     \__stex_smsmode_do:w
1725   }
1726 }
1727 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1728   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1729     \expandafter\if\expandafter\relax\noexpand#1
1730     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1731     \else\expandafter\__stex_smsmode_do:w\fi
1732   }{
1733     \__stex_smsmode_do:w % #1
1734   }
1735 }
1736 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1737   \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1738     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1739       #1\__stex_smsmode_do:w
1740     }{
1741       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1742         #1
1743       }{
1744         \cs_if_eq:NNTF \begin #1 {
1745           \__stex_smsmode_check_begin:n
1746         }{
1747           \cs_if_eq:NNTF \end #1 {
1748             \__stex_smsmode_check_end:n

```

```

1749         }{
1750         \__stex_smsmode_do:w
1751         }
1752     }
1753 }
1754 }
1755 }
1756 }
1757
1758 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1759 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1760 \begin{#1}
1761 }{
1762 \__stex_smsmode_do:w
1763 }
1764 }
1765 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1766 \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1767 \end{#1}\__stex_smsmode_do:w
1768 }{
1769 \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1770 }
1771 }

```

(End definition for `\stex_smsmode_do:.` This function is documented on page 58.)

28.2 Inheritance

```

1772 <@@=stex_importmodule>

\stex_import_module_uri:nn

1773 \cs_new_protected:Nn \stex_import_module_uri:nn {
1774 \str_set:Nx \l_stex_import_archive_str { #1 }
1775 \str_set:Nn \l_stex_import_path_str { #2 }
1776
1777 \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1778 \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1779 \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1780
1781 \stex_modules_current_namespace:
1782 \bool_lazy_all:nTF {
1783 {\str_if_empty_p:N \l_stex_import_archive_str}
1784 {\str_if_empty_p:N \l_stex_import_path_str}
1785 {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1786 }{
1787 \str_set_eq:NN \l_stex_import_path_str \l_stex_module_subpath_str
1788 \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1789 }{
1790 \str_if_empty:NT \l_stex_import_archive_str {
1791 \prop_if_exist:NT \l_stex_current_repository_prop {
1792 \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1793 }
1794 }
1795 \str_if_empty:NTF \l_stex_import_archive_str {

```

```

1796     \str_if_empty:NF \l_stex_import_path_str {
1797         \str_set:Nx \l_stex_import_ns_str {
1798             \l_stex_module_ns_str / \l_stex_import_path_str
1799         }
1800     }
1801 }{
1802     \stex_require_repository:n \l_stex_import_archive_str
1803     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1804     \l_stex_import_ns_str
1805     \str_if_empty:NF \l_stex_import_path_str {
1806         \str_set:Nx \l_stex_import_ns_str {
1807             \l_stex_import_ns_str / \l_stex_import_path_str
1808         }
1809     }
1810 }
1811 }
1812 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 59.)

```

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

```

(End definition for `\l_stex_import_name_str` and others. These variables are documented on page 59.)

```

\stex_import_require_module:nnnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1817 \cs_new_protected:Nn \stex_import_require_module:nnnnn {
1818     \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1819
1820         %\stex_debug:nn{requiremodule}{Here:\\~1::~~1\\~2::~~2\\~3::~~3\\~4::~~4}
1821
1822         \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/}} {#4}
1823         \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1824
1825         %\stex_debug:nn{requiremodule}{Top~module:\l_tmpc_str}
1826
1827         % archive
1828         \str_set:Nx \l_tmpa_str { #2 }
1829         \str_if_empty:NTF \l_tmpa_str {
1830             \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1831         } {
1832             \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1833             \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1834             \seq_put_right:Nn \l_tmpa_seq { source }
1835         }
1836
1837         % path
1838         \str_set:Nx \l_tmpb_str { #3 }
1839         \str_if_empty:NTF \l_tmpb_str {
1840             \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1841

```

```

1842 \ltx@ifpackageloaded{babel} {
1843   \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1844     { \language } \l_tmpb_str {
1845       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1846     }
1847 } {
1848   \str_clear:N \l_tmpb_str
1849 }
1850
1851 %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1852 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1853   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1854 }{
1855   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1856   \IfFileExists{ \l_tmpa_str.tex }{
1857     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1858   }{
1859     % try english as default
1860     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1861     \IfFileExists{ \l_tmpa_str.en.tex }{
1862       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1863     }{
1864       \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1865     }
1866   }
1867 }
1868
1869 } {
1870   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1871   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1872
1873   \ltx@ifpackageloaded{babel} {
1874     \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1875       { \language } \l_tmpb_str {
1876         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1877       }
1878   } {
1879     \str_clear:N \l_tmpb_str
1880   }
1881
1882   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1883
1884   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1885   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1886     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1887   }{
1888     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1889     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1890       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1891     }{
1892       % try english as default
1893       %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1894       \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1895         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }

```

```

1896     }{
1897       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1898       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1899         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1900       }{
1901         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1902         \IfFileExists{ \l_tmpa_str.tex }{
1903           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1904         }{
1905           % try english as default
1906           %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1907           \IfFileExists{ \l_tmpa_str.en.tex }{
1908             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1909           }{
1910             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1911           }
1912         }
1913       }
1914     }
1915   }
1916 }
1917 }
1918
1919 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1920   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1921     \seq_clear:N \l_stex_all_modules_seq
1922     \str_clear:N \l_stex_current_module_str
1923     \str_set:Nx \l_tmpb_str { #2 }
1924     \str_if_empty:NF \l_tmpb_str {
1925       \stex_set_current_repository:n { #2 }
1926     }
1927     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1928   }
1929
1930   \stex_if_module_exists:nF { #1 ? #4 } {
1931     \msg_error:nnx{stex}{error/unknownmodule}{
1932       #1?#4~(in~file~\g__stex_importmodule_file_str)
1933     }
1934   }
1935 }
1936
1937 }
1938 \stex_activate_module:n { #1 ? #4 }
1939 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 59.)

`\importmodule`

```

1940 \NewDocumentCommand \importmodule { 0{} m } {
1941   \stex_import_module_uri:nn { #1 } { #2 }
1942   \stex_debug:nn{modules}{Importing~module:~
1943     \l_stex_import_ns_str ? \l_stex_import_name_str
1944   }
1945   \stex_import_require_module:nnnn

```

```

1946 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1947 { \l_stex_import_path_str } { \l_stex_import_name_str }
1948 \stex_if_smsmode:F {
1949   \stex_annotate_invisible:nnn
1950   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1951 }
1952 \exp_args:Nx \stex_add_to_current_module:n {
1953   \stex_import_require_module:nnnn
1954   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1955   { \l_stex_import_path_str } { \l_stex_import_name_str }
1956 }
1957 \exp_args:Nx \stex_add_import_to_current_module:n {
1958   \l_stex_import_ns_str ? \l_stex_import_name_str
1959 }
1960 \stex_smsmode_do:
1961 \ignorespacesandpars
1962 }
1963 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page 58.)

`\usemodule`

```

1964 \NewDocumentCommand \usemodule { 0{} m } {
1965   \stex_if_smsmode:F {
1966     \stex_import_module_uri:nn { #1 } { #2 }
1967     \stex_import_require_module:nnnn
1968     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1969     { \l_stex_import_path_str } { \l_stex_import_name_str }
1970     \stex_annotate_invisible:nnn
1971     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1972   }
1973   \stex_smsmode_do:
1974   \ignorespacesandpars
1975 }

```

(End definition for `\usemodule`. This function is documented on page 58.)

```

1976 </package>

```

Chapter 29

STEX -Symbols Implementation

```
1977 <*package>
1978
1979 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1980
      Warnings and error messages
1981 \msg_new:nnn{stex}{error/wrongargs}{
1982   args~value~in~symbol~declaration~for~#1~
1983   needs~to~be~i,~a,~b~or~B,~but~#2~given
1984 }
1985 \msg_new:nnn{stex}{error/unknownsymbol}{
1986   No~symbol~#1~found!
1987 }
1988 \msg_new:nnn{stex}{error/seqlength}{
1989   Expected~#1~arguments;~got~#2!
1990 }
1991 \msg_new:nnn{stex}{error/unknownnotation}{
1992   Unknown~notation~#1~for~#2!
1993 }
```

29.1 Symbol Declarations

```
1994 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1995 \cs_new_protected:Nn \stex_all_symbols:n {
1996   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1997   \seq_map_inline:Nn \l_stex_all_modules_seq {
1998     \seq_map_inline:cn{c_stex_module_##1_constants}{
1999       \__stex_symdecl_all_symbols_cs{##1?####1}
2000     }
2001   }
2002 }
```

(End definition for `\stex_all_symbols:n`. This function is documented on page 61.)

`\STEXsymbol`

```
2003 \NewDocumentCommand \STEXsymbol { m } {
2004   \stex_get_symbol:n { #1 }
2005   \exp_args:No
2006   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
2007 }
```

(End definition for `\STEXsymbol`. This function is documented on page 62.)

`symdecl` arguments:

```
2008 \keys_define:nn { stex / symdecl } {
2009   name          .str_set_x:N = \l_stex_symdecl_name_str ,
2010   local         .bool_set:N = \l_stex_symdecl_local_bool ,
2011   args          .str_set_x:N = \l_stex_symdecl_args_str ,
2012   type          .tl_set:N = \l_stex_symdecl_type_tl ,
2013   deprecate     .str_set_x:N = \l_stex_symdecl_deprecate_str ,
2014   align         .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
2015   gfc           .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
2016   specializes   .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
2017   def           .tl_set:N = \l_stex_symdecl_definiens_tl ,
2018   assoc         .choices:nn =
2019     {bin,binl,binr,pre,conj,pwconj}
2020     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
2021 }
2022
2023 \bool_new:N \l_stex_symdecl_make_macro_bool
2024
2025 \cs_new_protected:Nn \__stex_symdecl_args:n {
2026   \str_clear:N \l_stex_symdecl_name_str
2027   \str_clear:N \l_stex_symdecl_args_str
2028   \str_clear:N \l_stex_symdecl_deprecate_str
2029   \str_clear:N \l_stex_symdecl_assoctype_str
2030   \bool_set_false:N \l_stex_symdecl_local_bool
2031   \tl_clear:N \l_stex_symdecl_type_tl
2032   \tl_clear:N \l_stex_symdecl_definiens_tl
2033
2034   \keys_set:nn { stex / symdecl } { #1 }
2035 }
```

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

```
2036
2037 \NewDocumentCommand \symdecl { s m O{} } {
2038   \__stex_symdecl_args:n { #3 }
2039   \IfBooleanTF #1 {
2040     \bool_set_false:N \l_stex_symdecl_make_macro_bool
2041   } {
2042     \bool_set_true:N \l_stex_symdecl_make_macro_bool
2043   }
2044   \stex_symdecl_do:n { #2 }
2045   \stex_smsmode_do:
2046 }
2047
2048 \cs_new_protected:Nn \stex_symdecl_do:nn {
```



```

2049 \__stex_symdecl_args:n{#1}
2050 \bool_set_false:N \l_stex_symdecl_make_macro_bool
2051 \stex_symdecl_do:n{#2}
2052 }
2053
2054 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for \symdecl. This function is documented on page 60.)

\stex_symdecl_do:n

```

2055 \cs_new_protected:Nn \stex_symdecl_do:n {
2056   \str_if_in_module:F {
2057     % TODO throw error? some default namespace?
2058   }
2059
2060   \str_if_empty:NT \l_stex_symdecl_name_str {
2061     \str_set:Nx \l_stex_symdecl_name_str { #1 }
2062   }
2063
2064   \prop_if_exist:cT { l_stex_symdecl_
2065     \l_stex_current_module_str ?
2066     \l_stex_symdecl_name_str
2067   _prop
2068   }{
2069     % TODO throw error (beware of circular dependencies)
2070   }
2071
2072   \prop_clear:N \l_tmpa_prop
2073   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2074   \seq_clear:N \l_tmpa_seq
2075   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2076   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2077
2078   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2079     \str_if_empty:NF \l_stex_module_deprecate_str {
2080       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2081     }
2082   }
2083   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2084
2085   \exp_args:No \stex_add_constant_to_current_module:n {
2086     \l_stex_symdecl_name_str
2087   }
2088
2089   % arity/args
2090   \int_zero:N \l_tmpb_int
2091
2092   \bool_set_true:N \l_tmpa_bool
2093   \str_map_inline:Nn \l_stex_symdecl_args_str {
2094     \token_case_meaning:NnF ##1 {
2095       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2096       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2097       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2098       {\tl_to_str:n a} {

```

```

2099     \bool_set_false:N \l_tmpa_bool
2100     \int_incr:N \l_tmpb_int
2101   }
2102   {\tl_to_str:n B} {
2103     \bool_set_false:N \l_tmpa_bool
2104     \int_incr:N \l_tmpb_int
2105   }
2106   }{
2107     \msg_error:nnxx{stex}{error/wrongargs}{
2108       \l_stex_current_module_str ?
2109       \l_stex_symdecl_name_str
2110     }{##1}
2111   }
2112 }
2113 \bool_if:NTF \l_tmpa_bool {
2114   % possibly numeric
2115   \str_if_empty:NTF \l_stex_symdecl_args_str {
2116     \prop_put:Nnn \l_tmpa_prop { args } {}
2117     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2118   }{
2119     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2120     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2121     \str_clear:N \l_tmpa_str
2122     \int_step_inline:nn \l_tmpa_int {
2123       \str_put_right:Nn \l_tmpa_str i
2124     }
2125     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2126   }
2127 } {
2128   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2129   \prop_put:Nnx \l_tmpa_prop { arity }
2130     { \str_count:N \l_stex_symdecl_args_str }
2131 }
2132 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2133
2134 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2135   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2136 }{
2137   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2138 }
2139
2140 % semantic macro
2141
2142 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2143   \exp_args:Nx \stex_do_up_to_module:n {
2144     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2145       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2146     }}
2147   }
2148
2149   \bool_if:NF \l_stex_symdecl_local_bool {
2150     \exp_args:Nx \stex_add_to_current_module:n {
2151       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2152         \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

2153     } }
2154   }
2155 }
2156 }
2157
2158 \stex_debug:nn{symbols}{New~symbol:~
2159   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2160   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2161   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2162   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2163 }
2164
2165 % circular dependencies require this:
2166
2167 \prop_if_exist:cF {
2168   \l_stex_symdecl_
2169   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2170   _prop
2171 } {
2172   \exp_args:Nx \stex_do_up_to_module:n {
2173     \prop_set_from_keyval:cn {
2174       \l_stex_symdecl_
2175       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2176       _prop
2177     } {\prop_to_keyval:N \l_tmpa_prop}
2178     \seq_clear:c {
2179       \l_stex_symdecl_
2180       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2181       _notations
2182     }
2183   }
2184 }
2185
2186 \bool_if:NF \l_stex_symdecl_local_bool {
2187   \exp_args:Nx
2188   \stex_add_to_current_module:n {
2189     \seq_clear:c {
2190       \l_stex_symdecl_
2191       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2192       _notations
2193     }
2194     \prop_set_from_keyval:cn {
2195       \l_stex_symdecl_
2196       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2197       _prop
2198     } {
2199       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2200       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2201       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2202       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2203       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2204       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2205       defined   = \prop_item:Nn \l_tmpa_prop { defined }   ,
2206     }
  
```

```

2207     }
2208 }
2209
2210 \stex_if_smsmode:F {
2211 %   \exp_args:Nx \stex_do_up_to_module:n {
2212 %       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2213 %       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2214 %   }
2215 % }
2216 \stex_if_do_html:T {
2217   \stex_annotate_invisible:nnn {symdecl} {
2218     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2219   } {
2220     \tl_if_empty:NF \l_stex_symdecl_type_tl {
2221       \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2222     }
2223     \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$}
2224     \prop_item:Nn \l_tmpa_prop { args }
2225   }
2226   \stex_annotate_invisible:nnn{macroname}{\l_stex_symdecl_type_tl$}
2227   \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2228     \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2229   }
2230 }
2231 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2232   \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}
2233 }
2234 }
2235 }
2236 }
2237 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 61.)

`\stex_get_symbol:n`

```

2238 \str_new:N \l_stex_get_symbol_uri_str
2239
2240 \cs_new_protected:Nn \stex_get_symbol:n {
2241   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2242     \tl_set:Nn \l_tmpa_tl { #1 }
2243     \__stex_symdecl_get_symbol_from_cs:
2244   }{
2245     % argument is a string
2246     % is it a command name?
2247     \cs_if_exist:cTF { #1 }{
2248       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2249       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2250       \str_if_empty:NNTF \l_tmpa_str {
2251         \exp_args:Nx \cs_if_eq:NNTF {
2252           \tl_head:N \l_tmpa_tl
2253         } \stex_invoke_symbol:n {
2254           \__stex_symdecl_get_symbol_from_cs:
2255         }{
2256           \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2257     }
2258   } {
2259     \_stex_symdecl_get_symbol_from_string:n { #1 }
2260   }
2261   }{
2262     % argument is not a command name
2263     \_stex_symdecl_get_symbol_from_string:n { #1 }
2264     % \l_stex_all_symbols_seq
2265   }
2266 }
2267 \str_if_eq:eeF {
2268   \prop_item:cn {
2269     l_stex_symdecl\_l_stex_get_symbol_uri_str _prop
2270   }{ deprecate }
2271 }{}{
2272   \msg_warning:nnxx{stex}{warning/deprecated}{
2273     Symbol~\l_stex_get_symbol_uri_str
2274   }{
2275     \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{ deprecate }
2276   }
2277 }
2278 }
2279
2280 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_string:n {
2281   \tl_set:Nn \l_tmpa_tl {
2282     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2283   }
2284   \str_set:Nn \l_tmpa_str { #1 }
2285   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2286
2287   \stex_all_symbols:n {
2288     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2289       \seq_map_break:n{\seq_map_break:n{
2290         \tl_set:Nn \l_tmpa_tl {
2291           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2292         }
2293       }}
2294     }
2295   }
2296
2297   \l_tmpa_tl
2298 }
2299
2300 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2301   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2302   { \tl_tail:N \l_tmpa_tl }
2303   \tl_if_single:NTF \l_tmpa_tl {
2304     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2305       \exp_after:wN \str_set:Nn \exp_after:wN
2306       \l_stex_get_symbol_uri_str \l_tmpa_tl
2307     }{
2308       % TODO
2309       % tail is not a single group
2310     }

```

```

2311 }{
2312   % TODO
2313   % tail is not a single group
2314 }
2315 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page 61.)

29.2 Notations

```

2316 <@@=stex_notation>
      notation arguments:
2317 \keys_define:nn { stex / notation } {
2318   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2319   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2320   prec .str_set_x:N = \l__stex_notation_prec_str ,
2321   op .tl_set:N = \l__stex_notation_op_tl ,
2322   primary .bool_set:N = \l__stex_notation_primary_bool ,
2323   primary .default:n = {true} ,
2324   unknown .code:n = \str_set:Nx
2325     \l__stex_notation_variant_str \l_keys_key_str
2326 }
2327
2328 \cs_new_protected:Nn \_stex_notation_args:n {
2329   \str_clear:N \l__stex_notation_lang_str
2330   \str_clear:N \l__stex_notation_variant_str
2331   \str_clear:N \l__stex_notation_prec_str
2332   \tl_clear:N \l__stex_notation_op_tl
2333   \bool_set_false:N \l__stex_notation_primary_bool
2334
2335   \keys_set:nn { stex / notation } { #1 }
2336 }

```

`\notation`

```

2337 \NewDocumentCommand \notation { s m O{}} {
2338   \_stex_notation_args:n { #3 }
2339   \tl_clear:N \l_stex_symdecl_definiens_tl
2340   \stex_get_symbol:n { #2 }
2341   \tl_set:Nn \l_stex_notation_after_do_tl {
2342     \__stex_notation_final:
2343     \IfBooleanTF#1{
2344       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2345     }{}
2346     \stex_smsmode_do:\ignorespacesandpars
2347   }
2348   \stex_notation_do:nnnnn
2349   { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2350   { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2351   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2352   { \l__stex_notation_prec_str }
2353 }
2354 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for `\notation`. This function is documented on page 61.)

\stex_notation_do:nnnnn

```

2355 \seq_new:N \l__stex_notation_precedences_seq
2356 \tl_new:N \l__stex_notation_opprec_tl
2357 \int_new:N \l__stex_notation_currarg_int
2358 \tl_new:N \stex_symbol_after_invokation_tl
2359
2360 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2361   \let\l_stex_current_symbol_str\relax
2362   \seq_clear:N \l__stex_notation_precedences_seq
2363   \tl_clear:N \l__stex_notation_opprec_tl
2364   \str_set:Nx \l__stex_notation_args_str { #1 }
2365   \str_set:Nx \l__stex_notation_arity_str { #2 }
2366   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2367   \str_set:Nx \l__stex_notation_prec_str { #4 }
2368
2369   % precedences
2370   \str_if_empty:NTF \l__stex_notation_prec_str {
2371     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2372       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2373     }{
2374       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2375     }
2376   } {
2377     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2378       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2379       \int_step_inline:nn { \l__stex_notation_arity_str } {
2380         \exp_args:NNo
2381         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2382       }
2383     }{
2384       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2385       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2386         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2387         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2388           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2389             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2390           \seq_map_inline:Nn \l_tmpa_seq {
2391             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2392           }
2393         }
2394       }{
2395         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2396           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2397         }{
2398           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2399         }
2400       }
2401     }
2402   }
2403
2404   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2405   \int_step_inline:nn { \l__stex_notation_arity_str } {
2406     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2407       \exp_args:NNo

```

```

2408     \seq_put_right:No \l__stex_notation_precedences_seq {
2409         \l__stex_notation_opprec_tl
2410     }
2411 }
2412 }
2413 \tl_clear:N \l_stex_notation_dummyargs_tl
2414
2415 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2416     \exp_args:NNe
2417     \cs_set:Npn \l_stex_notation_macrocode_cs {
2418         \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2419         { \l__stex_notation_suffix_str }
2420         { \l__stex_notation_opprec_tl }
2421         { \exp_not:n { #5 } }
2422     }
2423     \l_stex_notation_after_do_tl
2424 }{
2425     \str_if_in:NnTF \l__stex_notation_args_str b {
2426         \exp_args:Nne \use:nn
2427         {
2428             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2429             \cs_set:Npn \l__stex_notation_arity_str } { {
2430                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2431                 { \l__stex_notation_suffix_str }
2432                 { \l__stex_notation_opprec_tl }
2433                 { \exp_not:n { #5 } }
2434             } }
2435     }{
2436         \str_if_in:NnTF \l__stex_notation_args_str B {
2437             \exp_args:Nne \use:nn
2438             {
2439                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2440                 \cs_set:Npn \l__stex_notation_arity_str } { {
2441                     \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2442                     { \l__stex_notation_suffix_str }
2443                     { \l__stex_notation_opprec_tl }
2444                     { \exp_not:n { #5 } }
2445                 } }
2446     }{
2447         \exp_args:Nne \use:nn
2448         {
2449             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2450             \cs_set:Npn \l__stex_notation_arity_str } { {
2451                 \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2452                 { \l__stex_notation_suffix_str }
2453                 { \l__stex_notation_opprec_tl }
2454                 { \exp_not:n { #5 } }
2455             } }
2456     }
2457 }
2458
2459 \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2460 \int_zero:N \l__stex_notation_currarg_int
2461 \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq

```



```

2462     \__stex_notation_arguments:
2463   }
2464 }

```

(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

```

2465 \cs_new_protected:Nn \__stex_notation_arguments: {
2466   \int_incr:N \l__stex_notation_currarg_int
2467   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2468     \l_stex_notation_after_do_tl
2469   }{
2470     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2471     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2472     \str_if_eq:NnTF \l_tmpa_str a {
2473       \__stex_notation_argument_assoc:nn{a}
2474     }{
2475       \str_if_eq:NnTF \l_tmpa_str B {
2476         \__stex_notation_argument_assoc:nn{B}
2477       }{
2478         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2479         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2480           { \_stex_term_math_arg:nnn
2481             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2482             { \l_tmpb_str }
2483             { ###\int_use:N \l__stex_notation_currarg_int }
2484           }
2485         }
2486         \__stex_notation_arguments:
2487       }
2488     }
2489   }
2490 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:nn

```

2491 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2492
2493   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2494     {\l__stex_notation_arity_str}{
2495     #2
2496   }
2497   \int_zero:N \l_tmpa_int
2498   \tl_clear:N \l_tmpa_tl
2499   \str_map_inline:Nn \l__stex_notation_args_str {
2500     \int_incr:N \l_tmpa_int
2501     \tl_put_right:Nx \l_tmpa_tl {
2502       \str_if_eq:nnTF {##1}{a}{ {} }{
2503         \str_if_eq:nnTF {##1}{B}{ {} }{
2504           {\_stex_term_arg:nn{##1}\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2505         }
2506       }
2507     }

```

```

2508 }
2509 \exp_after:wN\exp_after:wN\exp_after:wN \def
2510 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2511 \exp_after:wN\exp_after:wN\exp_after:wN ##
2512 \exp_after:wN\exp_after:wN\exp_after:wN 1
2513 \exp_after:wN\exp_after:wN\exp_after:wN ##
2514 \exp_after:wN\exp_after:wN\exp_after:wN 2
2515 \exp_after:wN\exp_after:wN\exp_after:wN {
2516   \exp_after:wN \exp_after:wN \exp_after:wN
2517   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2518     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2519   }
2520 }
2521
2522 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2523 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2524   \stex_term_math_assoc_arg:nnnn
2525   { #1\int_use:N \l__stex_notation_currarg_int }
2526   { \l_tmpa_str }
2527   { ####\int_use:N \l__stex_notation_currarg_int }
2528   { \l_tmpa_cs {####1} {####2} }
2529 } }
2530 \__stex_notation_arguments:
2531 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2532 \cs_new_protected:Nn \__stex_notation_final: {
2533 % \exp_args:Nne \use:nn
2534 % {
2535 % \cs_generate_from_arg_count:cNnn {
2536 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2537 %   \l__stex_notation_suffix_str
2538 %   _cs
2539 % }
2540 % \cs_set:Npn \l__stex_notation_arity_str } { {
2541 %   \exp_after:wN \exp_after:wN \exp_after:wN
2542 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2543 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
2544 % } }
2545
2546 % \tl_if_empty:NF \l__stex_notation_op_tl {
2547 %   \cs_set:cpx {
2548 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2549 %     \l__stex_notation_suffix_str
2550 %     _cs
2551 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2552 % }
2553
2554 \exp_args:Nx \stex_do_up_to_module:n {
2555   \cs_generate_from_arg_count:cNnn {
2556     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2557     \l__stex_notation_suffix_str

```

```

2558     _cs
2559   } \cs_set:Npn {\l__stex_notation_arity_str} {
2560     \exp_after:wN \exp_after:wN \exp_after:wN
2561     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2562     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2563   }
2564   \tl_if_empty:NF \l__stex_notation_op_tl {
2565     \cs_set:cpn {
2566       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2567       \l__stex_notation_suffix_str
2568       _cs
2569     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2570   }
2571 }
2572
2573 \exp_args:Ne
2574 \stex_add_to_current_module:n {
2575   \cs_generate_from_arg_count:cNnn {
2576     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2577     \l__stex_notation_suffix_str
2578     _cs
2579   } \cs_set:Npn {\l__stex_notation_arity_str} {
2580     \exp_after:wN \exp_after:wN \exp_after:wN
2581     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2582     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2583   }
2584   \tl_if_empty:NF \l__stex_notation_op_tl {
2585     \cs_set:cpn {
2586       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2587       \l__stex_notation_suffix_str
2588       _cs
2589     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2590   }
2591 }
2592
2593 \stex_debug:nn{symbols}{
2594   Notation~\l__stex_notation_suffix_str
2595   ~for~\l_stex_get_symbol_uri_str^^J
2596   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2597   Argument~precedences:~
2598     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2599   Notation: \cs_meaning:c {
2600     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2601     \l__stex_notation_suffix_str
2602     _cs
2603   }
2604 }
2605
2606 \exp_args:Ne
2607 \stex_do_up_to_module:n {
2608   \exp_not:N \seq_if_exist:cT { l_stex_symdecl_\l_stex_get_symbol_uri_str _notations }{
2609     \seq_put_right:cn {
2610       l_stex_symdecl_\l_stex_get_symbol_uri_str
2611       _notations

```

```

2612     } {\l__stex_notation_suffix_str}
2613   }
2614 }
2615 \exp_args:Ne
2616 \stex_add_to_current_module:n {
2617   \seq_put_right:cn {
2618     l_stex_symdecl_l\l_stex_get_symbol_uri_str
2619     _notations
2620   } { \l__stex_notation_suffix_str }
2621 }
2622
2623 \stex_if_smsmode:F {
2624
2625   % HTML annotations
2626   \stex_if_do_html:T {
2627     \stex_annotate_invisible:nnn { notation }
2628     { \l_stex_get_symbol_uri_str } {
2629       \stex_annotate_invisible:nnn { notationfragment }
2630       { \l__stex_notation_suffix_str }{}
2631       \stex_annotate_invisible:nnn { precedence }
2632       { \l__stex_notation_prec_str }{}
2633
2634       \int_zero:N \l_tmpa_int
2635       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2636       \tl_clear:N \l_tmpa_tl
2637       \int_step_inline:nn { \l__stex_notation_arity_str }{
2638         \int_incr:N \l_tmpa_int
2639         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2640         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2641         \str_if_eq:VnTF \l_tmpb_str a {
2642           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2643             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2644             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2645           } }
2646         }{
2647           \str_if_eq:VnTF \l_tmpb_str B {
2648             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2649               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2650               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2651             } }
2652           }{
2653             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2654               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2655             } }
2656           }
2657         }
2658       }
2659       \stex_annotate_invisible:nnn { notationcomp }{}{
2660         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2661         $ \exp_args:Nno \use:nn { \use:c {
2662           stex_notation_ \l_stex_current_symbol_str
2663           \c_hash_str \l__stex_notation_suffix_str _cs
2664         } } { \l_tmpa_tl } $
2665       }

```

```

2666     }
2667   }
2668 }
2669 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

2670 \keys_define:nn { stex / setnotation } {
2671   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2672   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2673   unknown .code:n = \str_set:Nx
2674     \l__stex_notation_variant_str \l_keys_key_str
2675 }
2676
2677 \cs_new_protected:Nn \stex_setnotation_args:n {
2678   \str_clear:N \l__stex_notation_lang_str
2679   \str_clear:N \l__stex_notation_variant_str
2680   \keys_set:nn { stex / setnotation } { #1 }
2681 }
2682
2683 \cs_new_protected:Nn \stex_setnotation:n {
2684   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2685     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2686     \exp_args:Nx \stex_do_up_to_module:n {
2687       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2688         \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2689         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2690         \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2691         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2692       }
2693     }
2694     \exp_args:Nx \stex_add_to_current_module:n {
2695       \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2696       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2697       \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2698       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2699     }
2700     \stex_debug:nn {notations}{
2701       Setting~default~notation~
2702       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2703       #1 \\
2704       \expandafter\meaning\csname
2705         l_stex_symdecl_#1 _notations\endcsname
2706     }
2707   }{
2708     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str \c_hash_str \l__s
2709   }
2710 }
2711
2712 \NewDocumentCommand \setnotation {m m} {
2713   \stex_get_symbol:n { #1 }
2714   \stex_setnotation_args:n { #2 }
2715   \stex_setnotation:n{\l_stex_get_symbol_uri_str}

```

```

2716 \stex_smsmode_do:\ignorespacesandpars
2717 }
2718
2719 \cs_new_protected:Nn \stex_copy_notations:nn {
2720   \stex_debug:nn {notations}{
2721     Copying~notations~from~#2~to~#1\\
2722     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2723   }
2724   \tl_clear:N \l_tmpa_tl
2725   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2726     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2727   }
2728   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2729     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2730     \edef \l_tmpa_tl {
2731       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2732       \exp_after:wN\exp_after:wN\exp_after:wN {
2733         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2734       }
2735     }
2736     \exp_args:Nx
2737     \stex_add_to_current_module:n {
2738       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2739         \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2740         \cs_generate_from_arg_count:cNnn {
2741           stex_notation_ #1 \c_hash_str ##1 _cs
2742         } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2743           \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2744         }
2745         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2746           \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2747             {\exp_args:NNo\exp_args:No\exp_not:n{\cename stex_op_notation_ #2\c_hash_str ##1
2748             }
2749         }
2750       }
2751       \exp_args:Nx
2752       \stex_do_up_to_module:n {
2753         \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2754           \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2755           \cs_generate_from_arg_count:cNnn {
2756             stex_notation_ #1 \c_hash_str ##1 _cs
2757           } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2758             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2759           }
2760           \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2761             \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2762               {\exp_args:NNo\exp_args:No\exp_not:n{\cename stex_op_notation_ #2\c_hash_str ##1
2763               }
2764           }
2765         }
2766       }
2767     }
2768
2769 \NewDocumentCommand \copynotation {m m} {

```

```

2770 \stex_get_symbol:n { #1 }
2771 \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2772 \stex_get_symbol:n { #2 }
2773 \exp_args:Noo
2774 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2775 \exp_args:Nx \stex_add_to_current_module:n{
2776   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2777 }
2778 \stex_smsmode_do:\ignorespacesandpars
2779 }
2780

```

(End definition for \setnotation. This function is documented on page 18.)

\symdef

```

2781 \keys_define:nn { stex / symdef } {
2782   name .str_set_x:N = \l_stex_symdecl_name_str ,
2783   local .bool_set:N = \l_stex_symdecl_local_bool ,
2784   args .str_set_x:N = \l_stex_symdecl_args_str ,
2785   type .tl_set:N = \l_stex_symdecl_type_tl ,
2786   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2787   op .tl_set:N = \l__stex_notation_op_tl ,
2788   lang .str_set_x:N = \l__stex_notation_lang_str ,
2789   variant .str_set_x:N = \l__stex_notation_variant_str ,
2790   prec .str_set_x:N = \l__stex_notation_prec_str ,
2791   assoc .choices:nn =
2792     {bin,binl,binr,pre,conj,pwconj}
2793     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2794   unknown .code:n = \str_set:Nx
2795     \l__stex_notation_variant_str \l_keys_key_str
2796 }
2797
2798 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2799   \str_clear:N \l_stex_symdecl_name_str
2800   \str_clear:N \l_stex_symdecl_args_str
2801   \str_clear:N \l_stex_symdecl_assoctype_str
2802   \bool_set_false:N \l_stex_symdecl_local_bool
2803   \tl_clear:N \l_stex_symdecl_type_tl
2804   \tl_clear:N \l_stex_symdecl_definiens_tl
2805   \str_clear:N \l__stex_notation_lang_str
2806   \str_clear:N \l__stex_notation_variant_str
2807   \str_clear:N \l__stex_notation_prec_str
2808   \tl_clear:N \l__stex_notation_op_tl
2809
2810   \keys_set:nn { stex / symdef } { #1 }
2811 }
2812
2813 \NewDocumentCommand \symdef { m O{} } {
2814   \__stex_notation_symdef_args:n { #2 }
2815   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2816   \stex_symdecl_do:n { #1 }
2817   \tl_set:Nn \l_stex_notation_after_do_tl {
2818     \__stex_notation_final:
2819     \stex_smsmode_do:\ignorespacesandpars

```

```

2820 }
2821 \str_set:Nx \l_stex_get_symbol_uri_str {
2822   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2823 }
2824 \exp_args:Nx \stex_notation_do:nnnnn
2825   { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2826   { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2827   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2828   { \l__stex_notation_prec_str }
2829 }
2830 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 61.)

29.3 Variables

```

2831 <@@=stex_variables>
2832
2833 \keys_define:nn { stex / vardef } {
2834   name      .str_set_x:N = \l__stex_variables_name_str ,
2835   args      .str_set_x:N = \l__stex_variables_args_str ,
2836   type      .tl_set:N    = \l__stex_variables_type_tl ,
2837   def       .tl_set:N    = \l__stex_variables_def_tl ,
2838   op        .tl_set:N    = \l__stex_variables_op_tl ,
2839   prec      .str_set_x:N = \l__stex_variables_prec_str ,
2840   assoc     .choices:nn =
2841     {bin,binl,binr,pre,conj,pwconj}
2842     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2843   bind      .choices:nn =
2844     {forall,exists}
2845     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2846 }
2847
2848 \cs_new_protected:Nn \__stex_variables_args:n {
2849   \str_clear:N \l__stex_variables_name_str
2850   \str_clear:N \l__stex_variables_args_str
2851   \str_clear:N \l__stex_variables_prec_str
2852   \str_clear:N \l__stex_variables_assoctype_str
2853   \str_clear:N \l__stex_variables_bind_str
2854   \tl_clear:N \l__stex_variables_type_tl
2855   \tl_clear:N \l__stex_variables_def_tl
2856   \tl_clear:N \l__stex_variables_op_tl
2857
2858   \keys_set:nn { stex / vardef } { #1 }
2859 }
2860
2861 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2862   \__stex_variables_args:n {#2}
2863   \str_if_empty:NT \l__stex_variables_name_str {
2864     \str_set:Nx \l__stex_variables_name_str { #1 }
2865   }
2866   \prop_clear:N \l_tmpa_prop
2867   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2868

```



```

2869 \int_zero:N \l_tmpb_int
2870 \bool_set_true:N \l_tmpa_bool
2871 \str_map_inline:Nn \l__stex_variables_args_str {
2872   \token_case_meaning:NnF ##1 {
2873     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2874     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2875     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2876     {\tl_to_str:n a} {
2877       \bool_set_false:N \l_tmpa_bool
2878       \int_incr:N \l_tmpb_int
2879     }
2880     {\tl_to_str:n B} {
2881       \bool_set_false:N \l_tmpa_bool
2882       \int_incr:N \l_tmpb_int
2883     }
2884   }{
2885     \msg_error:nnxx{stex}{error/wrongargs}{
2886       variable~\l__stex_variables_name_str
2887     }{##1}
2888   }
2889 }
2890 \bool_if:NTF \l_tmpa_bool {
2891   % possibly numeric
2892   \str_if_empty:NTF \l__stex_variables_args_str {
2893     \prop_put:Nnn \l_tmpa_prop { args } {}
2894     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2895   }{
2896     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2897     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2898     \str_clear:N \l_tmpa_str
2899     \int_step_inline:nn \l_tmpa_int {
2900       \str_put_right:Nn \l_tmpa_str i
2901     }
2902     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2903     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2904   }
2905 } {
2906   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2907   \prop_put:Nnx \l_tmpa_prop { arity }
2908     { \str_count:N \l__stex_variables_args_str }
2909 }
2910 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2911 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2912
2913 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2914
2915 \tl_if_empty:NF \l__stex_variables_op_tl {
2916   \cs_set:cpx {
2917     stex_var_op_notation_ \l__stex_variables_name_str _cs
2918   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2919 }
2920
2921 \tl_set:Nn \l_stex_notation_after_do_tl {
2922   \exp_args:Nne \use:nn {

```

```

2923 \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2924 \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2925 } {{
2926 \exp_after:wN \exp_after:wN \exp_after:wN
2927 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2928 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2929 }}
2930 \stex_if_do_html:T {
2931 \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2932 \stex_annotate_invisible:nnn { precedence }
2933 { \l__stex_variables_prec_str }}{
2934 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2935 \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2936 \stex_annotate_invisible:nnn{macroname}{#1}{
2937 \tl_if_empty:NF \l__stex_variables_def_tl {
2938 \stex_annotate_invisible:nnn{definiens}{
2939 {\l__stex_variables_def_tl$}
2940 }
2941 \str_if_empty:NF \l__stex_variables_assoctype_str {
2942 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2943 }
2944 \str_if_empty:NF \l__stex_variables_bind_str {
2945 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{
2946 }
2947 \int_zero:N \l_tmpa_int
2948 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2949 \tl_clear:N \l_tmpa_tl
2950 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{{
2951 \int_incr:N \l_tmpa_int
2952 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2953 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2954 \str_if_eq:VnTF \l_tmpb_str a {
2955 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2956 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2957 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2958 } }
2959 }}{
2960 \str_if_eq:VnTF \l_tmpb_str B {
2961 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2962 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2963 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2964 } }
2965 }}{
2966 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2967 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{
2968 } }
2969 }
2970 }
2971 }
2972 \stex_annotate_invisible:nnn { notationcomp }{{
2973 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2974 $ \exp_args:Nno \use:nn { \use:c {
2975 stex_var_notation_\l__stex_variables_name_str _cs
2976 } } { \l_tmpa_tl } $

```

```

2977     }
2978   }
2979   }\ignorespacesandpars
2980 }
2981
2982 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2983 }
2984
2985 \cs_new:Nn \_stex_reset:N {
2986   \tl_if_exist:NTF #1 {
2987     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2988   }{
2989     \let \exp_not:N #1 \exp_not:N \undefined
2990   }
2991 }
2992
2993 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2994   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2995   \exp_args:Nnx \use:nn {
2996     % TODO
2997     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2998       #2
2999     }
3000   }{
3001     \_stex_reset:N \varnot
3002     \_stex_reset:N \vartype
3003     \_stex_reset:N \vardefi
3004   }
3005 }
3006
3007 \NewDocumentCommand \vardef { s } {
3008   \IfBooleanTF#1 {
3009     \__stex_variables_do_complex:nn
3010   }{
3011     \__stex_variables_do_simple:nnn
3012   }
3013 }
3014
3015 \NewDocumentCommand \svar { 0{} m }{
3016   \tl_if_empty:nTF {#1}{
3017     \str_set:Nn \l_tmpa_str { #2 }
3018   }{
3019     \str_set:Nn \l_tmpa_str { #1 }
3020   }
3021   \_stex_term_omv:nn {
3022     var://\l_tmpa_str
3023   }{
3024     \exp_args:Nnx \use:nn {
3025       \def\comp{\_varcomp}
3026       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
3027       \comp{ #2 }
3028     }{
3029       \_stex_reset:N \comp
3030       \_stex_reset:N \l_stex_current_symbol_str

```

```

3031     }
3032   }
3033 }
3034
3035
3036
3037 \keys_define:nn { stex / varseq } {
3038   name      .str_set_x:N = \l__stex_variables_name_str ,
3039   args      .int_set:N   = \l__stex_variables_args_int ,
3040   type      .tl_set:N    = \l__stex_variables_type_tl  ,
3041   mid       .tl_set:N    = \l__stex_variables_mid_tl   ,
3042   bind      .choices:nn =
3043     {forall,exists}
3044     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
3045 }
3046
3047 \cs_new_protected:Nn \__stex_variables_seq_args:n {
3048   \str_clear:N \l__stex_variables_name_str
3049   \int_set:Nn \l__stex_variables_args_int 1
3050   \tl_clear:N \l__stex_variables_type_tl
3051   \str_clear:N \l__stex_variables_bind_str
3052
3053   \keys_set:nn { stex / varseq } { #1 }
3054 }
3055
3056 \NewDocumentCommand \varseq {m O{}} m m m){
3057   \__stex_variables_seq_args:n { #2 }
3058   \str_if_empty:NT \l__stex_variables_name_str {
3059     \str_set:Nx \l__stex_variables_name_str { #1 }
3060   }
3061   \prop_clear:N \l_tmpa_prop
3062   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
3063
3064   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3065   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3066     \msg_error:nnxx{stex}{error/seqlength}
3067     {\int_use:N \l__stex_variables_args_int}
3068     {\seq_count:N \l_tmpa_seq}
3069   }
3070   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3071   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3072     \msg_error:nnxx{stex}{error/seqlength}
3073     {\int_use:N \l__stex_variables_args_int}
3074     {\seq_count:N \l_tmpb_seq}
3075   }
3076   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3077   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3078
3079   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3080   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3081
3082   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3083   \int_step_inline:nn \l__stex_variables_args_int {
3084     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }

```

```

3085 }
3086 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3087 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3088 \tl_if_empty:NF \l__stex_variables_mid_tl {
3089   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3090   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3091 }
3092 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3093 \int_step_inline:nn \l__stex_variables_args_int {
3094   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3095 }
3096 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3097 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3098
3099
3100 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3101
3102 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3103
3104 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3105
3106 \int_step_inline:nn \l__stex_variables_args_int {
3107   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3108     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3109   }}
3110 }
3111
3112 \tl_set:Nx \l_tmpa_tl {
3113   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3114     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3115   }
3116 }
3117
3118 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3119
3120 \exp_args:Nno \use:nn {
3121   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3122   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3123
3124   \stex_debug:nn{sequences}{New~Sequence:~
3125     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str_cs\endcsname\\~\\
3126     \prop_to_keyval:N \l_tmpa_prop
3127   }
3128   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3129     \tl_if_empty:NF \l__stex_variables_type_tl {
3130       \stex_annotate:nnn {type}{}{\seqtype\l__stex_variables_type_tl$}
3131     }
3132     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3133     \str_if_empty:NF \l__stex_variables_bind_str {
3134       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3135     }
3136   }}
3137
3138   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str_prop}\l_tmpa_prop

```

```
3139 \ignorespacesandpars
3140 }
3141
3142 </package>
```

Chapter 30

STEX -Terms Implementation

```
3143 <*package>
3144
3145 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3146
3147 <@@=stex_terms>
3148
3149 Warnings and error messages
3150 \msg_new:nnn{stex}{error/nonotation}{
3151   Symbol~#1~invoked,~but~has~no~notation#2!
3152 }
3153 \msg_new:nnn{stex}{error/notationarg}{
3154   Error~in~parsing~notation~#1
3155 }
3156 \msg_new:nnn{stex}{error/noop}{
3157   Symbol~#1~has~no~operator~notation~for~notation~#2
3158 }
3159 \msg_new:nnn{stex}{error/notallowed}{
3160   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3161 }
3162 \msg_new:nnn{stex}{error/doubleargument}{
3163   Argument~#1~of~symbol~#2~already~assigned
3164 }
3165 \msg_new:nnn{stex}{error/overarity}{
3166   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3167 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3167
3168
3169 \bool_new:N \l_stex_allow_semantic_bool
3170 \bool_set_true:N \l_stex_allow_semantic_bool
3171
```

```

3172 \cs_new_protected:Nn \stex_invoke_symbol:n {
3173   \bool_if:NTF \l_stex_allow_semantic_bool {
3174     \str_if_eq:eeF {
3175       \prop_item:cn {
3176         l_stex_symdecl_#1_prop
3177       }{ deprecate }
3178     }{}{
3179       \msg_warning:nxxx{stex}{warning/deprecated}{
3180         Symbol~#1
3181       }{
3182         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3183       }
3184     }
3185     \if_mode_math:
3186       \exp_after:wN \__stex_terms_invoke_math:n
3187     \else:
3188       \exp_after:wN \__stex_terms_invoke_text:n
3189     \fi: { #1 }
3190   }{
3191     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3192   }
3193 }
3194
3195 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3196   \peek_charcode_remove:NTF ! {
3197     \__stex_terms_invoke_op_custom:nn {#1}
3198   }{
3199     \__stex_terms_invoke_custom:nn {#1}
3200   }
3201 }
3202
3203 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3204   \peek_charcode_remove:NTF ! {
3205     % operator
3206     \peek_charcode_remove:NTF * {
3207       % custom op
3208       \__stex_terms_invoke_op_custom:nn {#1}
3209     }{
3210       % op notation
3211       \peek_charcode:NTF [ {
3212         \__stex_terms_invoke_op_notation:nw {#1}
3213       }{
3214         \__stex_terms_invoke_op_notation:nw {#1}[]
3215       }
3216     }
3217   }{
3218     \peek_charcode_remove:NTF * {
3219       \__stex_terms_invoke_custom:nn {#1}
3220       % custom
3221     }{
3222       % normal
3223       \peek_charcode:NTF [ {
3224         \__stex_terms_invoke_notation:nw {#1}
3225       }{

```



```

3226     \__stex_terms_invoke_notation:nw {#1}[]
3227   }
3228 }
3229 }
3230 }
3231
3232
3233 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3234   \exp_args:Nnx \use:nn {
3235     \def\comp{\_comp}
3236     \str_set:Nn \l_stex_current_symbol_str { #1 }
3237     \bool_set_false:N \l_stex_allow_semantic_bool
3238     \stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3239       \comp{ #2 }
3240     }
3241   }{
3242     \stex_reset:N \comp
3243     \stex_reset:N \l_stex_current_symbol_str
3244     \bool_set_true:N \l_stex_allow_semantic_bool
3245   }
3246 }
3247
3248 \keys_define:nn { stex / terms } {
3249   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3250   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3251   unknown .code:n = \str_set:Nx
3252     \l_stex_notation_variant_str \l_keys_key_str
3253 }
3254
3255 \cs_new_protected:Nn \__stex_terms_args:n {
3256   \str_clear:N \l_stex_notation_lang_str
3257   \str_clear:N \l_stex_notation_variant_str
3258
3259   \keys_set:nn { stex / terms } { #1 }
3260 }
3261
3262 \cs_new_protected:Nn \stex_find_notation:nn {
3263   \__stex_terms_args:n { #2 }
3264   \seq_if_empty:cTF {
3265     l_stex_symdecl_ #1 _notations
3266   } {
3267     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3268   } {
3269     \bool_lazy_all:nTF {
3270       {\str_if_empty_p:N \l_stex_notation_variant_str}
3271       {\str_if_empty_p:N \l_stex_notation_lang_str}
3272     }{
3273       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3274     }{
3275       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3276         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3277       }{
3278         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3279       }{

```

```

3280         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3281         ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3282         }
3283     }
3284 }
3285 }
3286 }
3287
3288 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3289     \exp_args:Nnx \use:nn {
3290         \def\comp{\_comp}
3291         \str_set:Nn \l_stex_current_symbol_str { #1 }
3292         \stex_find_notation:nn { #1 }{ #2 }
3293         \bool_set_false:N \l_stex_allow_semantic_bool
3294         \cs_if_exist:cTF {
3295             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3296         }{
3297             \_stex_term_oms:nnn { #1 }{
3298                 #1 \c_hash_str \l_stex_notation_variant_str
3299             }{
3300                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3301             }
3302         }{
3303             \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3304                 \cs_if_exist:cTF {
3305                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3306                 }{
3307                     \tl_set:Nx \stex_symbol_after_invokation_tl {
3308                         \_stex_reset:N \comp
3309                         \_stex_reset:N \stex_symbol_after_invokation_tl
3310                         \_stex_reset:N \l_stex_current_symbol_str
3311                         \bool_set_true:N \l_stex_allow_semantic_bool
3312                     }
3313                     \def\comp{\_comp}
3314                     \str_set:Nn \l_stex_current_symbol_str { #1 }
3315                     \bool_set_false:N \l_stex_allow_semantic_bool
3316                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3317                 }{
3318                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3319                     ~\l_stex_notation_variant_str
3320                     }
3321                 }
3322             }{
3323                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3324             }
3325         }
3326     }{
3327         \_stex_reset:N \comp
3328         \_stex_reset:N \l_stex_current_symbol_str
3329         \bool_set_true:N \l_stex_allow_semantic_bool
3330     }
3331 }
3332
3333 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3334 \stex_find_notation:nn { #1 }{ #2 }
3335 \cs_if_exist:cTF {
3336   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3337 }{
3338   \tl_set:Nx \stex_symbol_after_invokation_tl {
3339     \stex_reset:N \comp
3340     \stex_reset:N \stex_symbol_after_invokation_tl
3341     \stex_reset:N \l_stex_current_symbol_str
3342     \bool_set_true:N \l_stex_allow_semantic_bool
3343   }
3344   \def\comp{\_comp}
3345   \str_set:Nn \l_stex_current_symbol_str { #1 }
3346   \bool_set_false:N \l_stex_allow_semantic_bool
3347   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3348 }{
3349   \msg_error:nxxx{stex}{error/nonotation}{#1}{
3350     ~\l_stex_notation_variant_str
3351   }
3352 }
3353 }
3354
3355 \prop_new:N \l__stex_terms_custom_args_prop
3356
3357 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3358   \exp_args:Nnx \use:nn {
3359     \bool_set_false:N \l_stex_allow_semantic_bool
3360     \def\comp{\_comp}
3361     \str_set:Nn \l_stex_current_symbol_str { #1 }
3362     \prop_clear:N \l__stex_terms_custom_args_prop
3363     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3364     \prop_get:cnN {
3365       l_stex_symdecl_#1 _prop
3366     }{ args } \l_tmpa_str
3367     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3368     \tl_set:Nn \arg { \__stex_terms_arg: }
3369     \str_if_empty:NTF \l_tmpa_str {
3370       \stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3371     }{
3372       \str_if_in:NnTF \l_tmpa_str b {
3373         \stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3374       }{
3375         \str_if_in:NnTF \l_tmpa_str B {
3376           \stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3377         }{
3378           \stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3379         }
3380       }
3381     }
3382     % TODO check that all arguments exist
3383   }{
3384     \stex_reset:N \l_stex_current_symbol_str
3385     \stex_reset:N \arg
3386     \stex_reset:N \comp
3387     \stex_reset:N \l__stex_terms_custom_args_prop

```

```

3388     \bool_set_true:N \l_stex_allow_semantic_bool
3389   }
3390 }
3391
3392 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3393   \tl_if_empty:nTF {#2}{
3394     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3395     \bool_set_true:N \l_tmpa_bool
3396     \bool_do_while:Nn \l_tmpa_bool {
3397       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3398       \int_incr:N \l_tmpa_int
3399     }{
3400       \bool_set_false:N \l_tmpa_bool
3401     }
3402   }
3403   ){
3404     \int_set:Nn \l_tmpa_int { #2 }
3405   }
3406   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3407   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3408     \msg_error:nnxxx{stex}{error/overarity}
3409     {\int_use:N \l_tmpa_int}
3410     {\l_stex_current_symbol_str}
3411     {\str_count:N \l_tmpa_str}
3412   }
3413   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3414   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3415     \bool_lazy_any:nF {
3416       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3417       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3418     }{
3419       \msg_error:nnxx{stex}{error/doubleargument}
3420       {\int_use:N \l_tmpa_int}
3421       {\l_stex_current_symbol_str}
3422     }
3423   }
3424   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3425   \bool_set_true:N \l_stex_allow_semantic_bool
3426   \IfBooleanTF#1{
3427     \stex_annotate_invisible:n { %TODO
3428       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3429     }
3430   }{ %TODO
3431     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3432   }
3433   \bool_set_false:N \l_stex_allow_semantic_bool
3434 }
3435
3436
3437 \cs_new_protected:Nn \_stex_term_arg:nn {
3438   \bool_set_true:N \l_stex_allow_semantic_bool
3439   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3440   \bool_set_false:N \l_stex_allow_semantic_bool
3441 }

```

```

3442
3443 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3444   \exp_args:Nnx \use:nn
3445     { \int_set:Nn \l__stex_terms_downprec { #2 }
3446       \_stex_term_arg:nn { #1 }{ #3 }
3447     }
3448   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3449   }

```

(End definition for `\stex_invoke_symbol:n`. This function is documented on page 62.)

`_stex_term_math_assoc_arg:nnnn`

```

3450 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3451   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3452   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3453   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3454     \expandafter\if\expandafter\relax\noexpand#3
3455     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3456     \else\expandafter\__stex_terms_math_assoc_arg_simple:nn
3457     \expandafter{\expandafter}\expandafter#3\fi
3458   }{
3459     \__stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3460   }
3461 }
3462
3463 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3464   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3465   \str_if_empty:NTF \l_tmpa_str {
3466     \exp_args:Nx \cs_if_eq:NNTF {
3467       \tl_head:N #1
3468     } \stex_invoke_sequence:n {
3469       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3470       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3471       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\l_tmpa_str _prop}{notation}}
3472       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3473       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3474         \exp_not:n{\exp_args:Nnx \use:nn} {
3475           \exp_not:n {
3476             \def\comp{\_varcomp}
3477             \str_set:Nn \l_stex_current_symbol_str
3478             } {varseq://\l_tmpa_str}
3479             \exp_not:n{ ##1 }
3480           }{
3481             \exp_not:n {
3482               \_stex_reset:N \comp
3483               \_stex_reset:N \l_stex_current_symbol_str
3484             }
3485           }
3486         }}}
3487       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3488       \seq_reverse:N \l_tmpa_seq
3489       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3490       \seq_map_inline:Nn \l_tmpa_seq {
3491         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {

```

```

3492         \exp_args:Nno
3493         \l_tmpa_cs { ##1 } \l_tmpa_tl
3494     }
3495 }
3496 \tl_set:Nx \l_tmpa_tl {
3497     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3498         \exp_args:No \exp_not:n \l_tmpa_tl
3499     }
3500 }
3501 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3502 }{
3503     \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3504 }
3505 } {
3506     \__stex_terms_math_assoc_arg_simple:nn{} { #1 }
3507 }
3508
3509 }
3510
3511 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:nn {
3512     \clist_set:Nn \l_tmpa_clist{ #2 }
3513     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3514         \tl_set:Nn \l_tmpa_tl { #2 }
3515     }{
3516         \clist_reverse:N \l_tmpa_clist
3517         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3518         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3519             \exp_args:No \exp_not:n \l_tmpa_tl
3520         }}
3521         \clist_map_inline:Nn \l_tmpa_clist {
3522             \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3523                 \exp_args:Nno
3524                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3525             }
3526         }
3527     }
3528     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3529 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 62.)

30.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
3530 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3531 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3532 \int_new:N \l__stex_terms_downprec
3533 \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 63.)

Bracketing:

\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str

```
3534 \tl_set:Nn \l__stex_terms_left_bracket_str (
3535 \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

```
3536 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3537   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3538     \bool_set_false:N \l__stex_terms_brackets_done_bool
3539     #2
3540   } {
3541     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3542       \bool_if:NTF \l__stex_inarray_bool { #2 }{
3543         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3544         \dobrackets { #2 }
3545       }
3546     }{ #2 }
3547   }
3548 }
```

(End definition for __stex_terms_maybe_brackets:nn.)

\dobrackets

```
3549 \bool_new:N \l__stex_terms_brackets_done_bool
3550 %\RequirePackage{scalerel}
3551 \cs_new_protected:Npn \dobrackets #1 {
3552   %\ThisStyle{\if D\m@switch
3553   %   \exp_args:Nnx \use:nn
3554   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3555   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3556   % \else
3557   \exp_args:Nnx \use:nn
3558   {
3559     \bool_set_true:N \l__stex_terms_brackets_done_bool
3560     \int_set:Nn \l__stex_terms_downprec \infprec
3561     \l__stex_terms_left_bracket_str
3562     #1
3563   }
3564   {
3565     \bool_set_false:N \l__stex_terms_brackets_done_bool
3566     \l__stex_terms_right_bracket_str
3567     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3568   }
3569   %\fi}
3570 }
```

(End definition for \dobrackets. This function is documented on page 63.)

\withbrackets

```
3571 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3572   \exp_args:Nnx \use:nn
3573   {
3574     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```

3575 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3576 #3
3577 }
3578 {
3579 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3580 {\l__stex_terms_left_bracket_str}
3581 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3582 {\l__stex_terms_right_bracket_str}
3583 }
3584 }

```

(End definition for `\withbrackets`. This function is documented on page 63.)

`\STEXinvisible`

```

3585 \cs_new_protected:Npn \STEXinvisible #1 {
3586 \stex_annotate_invisible:n { #1 }
3587 }

```

(End definition for `\STEXinvisible`. This function is documented on page 63.)

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3588 \cs_new_protected:Nn \_stex_term_oms:nnn {
3589 \stex_annotate:nnn{ OMID }{ #2 }{
3590 \stex_highlight_term:nn { #1 } { #3 }
3591 }
3592 }
3593
3594 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3595 \_stex_terms_maybe_brackets:nn { #3 }{
3596 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3597 }
3598 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 62.)

`_stex_term_math_omv:nn`

```

3599 \cs_new_protected:Nn \_stex_term_omv:nn {
3600 \stex_annotate:nnn{ OMV }{ #1 }{
3601 \stex_highlight_term:nn { #1 } { #2 }
3602 }
3603 }

```

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

`_stex_term_math_oma:nnnn`

```

3604 \cs_new_protected:Nn \_stex_term_oma:nnn {
3605 \stex_annotate:nnn{ OMA }{ #2 }{
3606 \stex_highlight_term:nn { #1 } { #3 }
3607 }
3608 }
3609
3610 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3611 \_stex_terms_maybe_brackets:nn { #3 }{
3612 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```



```

3613 }
3614 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 62.)

`_stex_term_math_omb:nnnn`

```

3615 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3616   \stex_annotate:nnn{ OMBIND }{ #2 }{
3617     \stex_highlight_term:nn { #1 } { #3 }
3618   }
3619 }
3620
3621 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3622   \__stex_terms_maybe_brackets:nn { #3 }{
3623     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3624   }
3625 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 62.)

`\symref`
`\symname`

```

3626 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3627
3628 \keys_define:nn { stex / symname } {
3629   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3630   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3631   root     .tl_set_x:N = \l__stex_terms_root_tl
3632 }
3633
3634 \cs_new_protected:Nn \stex_symname_args:n {
3635   \tl_clear:N \l__stex_terms_post_tl
3636   \tl_clear:N \l__stex_terms_pre_tl
3637   \tl_clear:N \l__stex_terms_root_str
3638   \keys_set:nn { stex / symname } { #1 }
3639 }
3640
3641 \NewDocumentCommand \symref { m m }{
3642   \let\compemph_uri_prev:\compemph@uri
3643   \let\compemph@uri\symrefemph@uri
3644   \STEXsymbol{#1}!{ #2 }
3645   \let\compemph@uri\compemph_uri_prev:
3646 }
3647
3648 \NewDocumentCommand \synonym { 0{} m m }{
3649   \stex_symname_args:n { #1 }
3650   \let\compemph_uri_prev:\compemph@uri
3651   \let\compemph@uri\symrefemph@uri
3652   % TODO
3653   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3654   \let\compemph@uri\compemph_uri_prev:
3655 }
3656
3657 \NewDocumentCommand \symname { 0{} m }{
3658   \stex_symname_args:n { #1 }
3659   \stex_get_symbol:n { #2 }

```

```

3660 \str_set:Nx \l_tmpa_str {
3661   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3662 }
3663 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3664
3665 \let\compemph_uri_prev:\compemph@uri
3666 \let\compemph@uri\symrefemph@uri
3667 \exp_args:NNx \use:nn
3668 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3669   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3670 } }
3671 \let\compemph@uri\compemph_uri_prev:
3672 }
3673
3674 \NewDocumentCommand \Symname { O{} m }{
3675   \stex_symname_args:n { #1 }
3676   \stex_get_symbol:n { #2 }
3677   \str_set:Nx \l_tmpa_str {
3678     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3679   }
3680   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3681   \let\compemph_uri_prev:\compemph@uri
3682   \let\compemph@uri\symrefemph@uri
3683   \exp_args:NNx \use:nn
3684   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3685     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3686     \l__stex_terms_post_tl
3687   } }
3688   \let\compemph@uri\compemph_uri_prev:
3689 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 62.)

30.3 Notation Components

```

3690 <@@=stex_notationcomps>
3691
\stex_highlight_term:nn
3691 \cs_new_protected:Nn \stex_highlight_term:nn {
3692   #2
3693 }
3694
3695 \cs_new_protected:Nn \stex_unhighlight_term:n {
3696   % \latexml_if:TF {
3697   %   #1
3698   % } {
3699   %   \rustex_if:TF {
3700   %     #1
3701   %   } {
3702     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3703   %   }
3704   % }
3705 }

```

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

```

\comp
\compemph@uri 3706 \cs_new_protected:Npn \_comp #1 {
\compemph 3707 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3708 \rustex_if:TF {
\defemph@uri 3709 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3710 }{
\symrefemph@uri 3711 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
\baremph 3712 }
\baremph@uri 3713 }
3714 }
3715
3716 \cs_new_protected:Npn \_varcomp #1 {
3717 \str_if_empty:NF \l_stex_current_symbol_str {
3718 \rustex_if:TF {
3719 \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3720 }{
3721 \exp_args:Nnx \baremph@uri { #1 } { \l_stex_current_symbol_str }
3722 }
3723 }
3724 }
3725
3726 \def\comp{\_comp}
3727
3728 \cs_new_protected:Npn \compemph@uri #1 #2 {
3729 \compemph{ #1 }
3730 }
3731
3732
3733 \cs_new_protected:Npn \compemph #1 {
3734 #1
3735 }
3736
3737 \cs_new_protected:Npn \defemph@uri #1 #2 {
3738 \defemph{#1}
3739 }
3740
3741 \cs_new_protected:Npn \defemph #1 {
3742 \textbf{#1}
3743 }
3744
3745 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3746 \symrefemph{#1}
3747 }
3748
3749 \cs_new_protected:Npn \symrefemph #1 {
3750 \textbf{#1}
3751 }
3752
3753 \cs_new_protected:Npn \baremph@uri #1 #2 {
3754 \baremph{#1}
3755 }
3756

```

```

3757 \cs_new_protected:Npn \varemp #1 {
3758   #1
3759 }

```

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

`\ellipses`

```

3760 \NewDocumentCommand \ellipses {} { \ldots }

```

(End definition for `\ellipses`. This function is documented on page 63.)

```

\parray
\prmatrix 3761 \bool_new:N \l_stex_inarray_bool
\parrayline 3762 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3763 \NewDocumentCommand \parray { m m } {
\parraycell 3764   \begingroup
3765   \bool_set_true:N \l_stex_inarray_bool
3766   \begin{array}{#1}
3767     #2
3768   \end{array}
3769   \endgroup
3770 }
3771
3772 \NewDocumentCommand \prmatrix { m } {
3773   \begingroup
3774   \bool_set_true:N \l_stex_inarray_bool
3775   \begin{matrix}
3776     #1
3777   \end{matrix}
3778   \endgroup
3779 }
3780
3781 \def \maybepline {
3782   \bool_if:NT \l_stex_inarray_bool {\hline}
3783 }
3784
3785 \def \parrayline #1 #2 {
3786   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3787 }
3788
3789 \def \pmrow #1 { \parrayline{}{ #1 } }
3790
3791 \def \parraylineh #1 #2 {
3792   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3793 }
3794
3795 \def \parraycell #1 {
3796   #1 \bool_if:NT \l_stex_inarray_bool {&}
3797 }

```

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

30.4 Variables

3798 <@@=stex_variables>

\stex_invoke_variable:n Invokes a variable

```

3799 \cs_new_protected:Nn \stex_invoke_variable:n {
3800   \if_mode_math:
3801     \exp_after:wN \__stex_variables_invoke_math:n
3802   \else:
3803     \exp_after:wN \__stex_variables_invoke_text:n
3804   \fi: {#1}
3805 }
3806
3807 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3808   %TODO
3809 }
3810
3811
3812 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3813   \peek_charcode_remove:NTF ! {
3814     \peek_charcode_remove:NTF ! {
3815       \peek_charcode:NTF [ {
3816         \__stex_variables_invoke_op_custom:nw
3817       }{
3818         % TODO throw error
3819       }
3820     }{
3821       \__stex_variables_invoke_op:n { #1 }
3822     }
3823   }{
3824     \peek_charcode_remove:NTF * {
3825       \__stex_variables_invoke_text:n { #1 }
3826     }{
3827       \__stex_variables_invoke_math_ii:n { #1 }
3828     }
3829   }
3830 }
3831
3832 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3833   \cs_if_exist:cTF {
3834     stex_var_op_notation_ #1 _cs
3835   }{
3836     \exp_args:Nnx \use:nn {
3837       \def\comp{\_varcomp}
3838       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3839       \stex_term_omv:nn { var://#1 }{
3840         \use:c{stex_var_op_notation_ #1 _cs }
3841       }
3842     }{
3843       \stex_reset:N \comp
3844       \stex_reset:N \l_stex_current_symbol_str
3845     }
3846   }{
3847     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{

```

```

3848     \stex_variables_invoke_math_ii:n {#1}
3849   }{
3850     \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
3851   }
3852 }
3853 }
3854
3855 \cs_new_protected:Npn \stex_variables_invoke_math_ii:n #1 {
3856   \cs_if_exist:cTF {
3857     stex_var_notation_#1_cs
3858   }{
3859     \tl_set:Nx \stex_symbol_after_invokation_tl {
3860       \stex_reset:N \comp
3861       \stex_reset:N \stex_symbol_after_invokation_tl
3862       \stex_reset:N \l_stex_current_symbol_str
3863       \bool_set_true:N \l_stex_allow_semantic_bool
3864     }
3865     \def\comp{\_varcomp}
3866     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3867     \bool_set_false:N \l_stex_allow_semantic_bool
3868     \use:c{stex_var_notation_#1_cs}
3869   }{
3870     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3871   }
3872 }

```

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

30.5 Sequences

```

3873 <@@=stex_sequences>
3874
3875 \cs_new_protected:Nn \stex_invoke_sequence:n {
3876   \peek_charcode_remove:NTF ! {
3877     \stex_term_omv:nn {varseq://#1}{
3878       \exp_args:Nnx \use:nn {
3879         \def\comp{\_varcomp}
3880         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3881         \prop_item:cn{stex_varseq_#1_prop}{notation}
3882       }{
3883         \stex_reset:N \comp
3884         \stex_reset:N \l_stex_current_symbol_str
3885       }
3886     }
3887   }{
3888     \bool_set_false:N \l_stex_allow_semantic_bool
3889     \def\comp{\_varcomp}
3890     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3891     \tl_set:Nx \stex_symbol_after_invokation_tl {
3892       \stex_reset:N \comp
3893       \stex_reset:N \stex_symbol_after_invokation_tl
3894       \stex_reset:N \l_stex_current_symbol_str
3895       \bool_set_true:N \l_stex_allow_semantic_bool
3896     }

```

```
3897     \use:c { stex_varseq_#1_cs }
3898   }
3899 }
3900 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
3901 ⟨*package⟩
3902
3903 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3904
3905
3906 Warnings and error messages
3907 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3908   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3909 }
3910 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
3911   Symbol~#1~not~assigned~in~interpretmodule~#2
3912 }
3913 \msg_new:nnn{stex}{error/unknownstructure}{
3914   No~structure~#1~found!
3915 }
3916 \msg_new:nnn{stex}{error/unknownfield}{
3917   No~field~#1~in~instance~#2~found!~\#3
3918 }
3919 \msg_new:nnn{stex}{error/keyval}{
3920   Invalid~key=value~pair~#1
3921 }
3922 \msg_new:nnn{stex}{error/instantiate/missing}{
3923   Assignments~missing~in~instantiate:~#1
3924 }
3925 \msg_new:nnn{stex}{error/incompatible}{
3926   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3927 }
3928
3929
```


31.1 Imports with modification

```

3930 <@@=stex_copymodule>
3931 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3932   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3933     \tl_set:Nn \l_tmpa_tl { #1 }
3934     \__stex_copymodule_get_symbol_from_cs:
3935   }{
3936     % argument is a string
3937     % is it a command name?
3938     \cs_if_exist:cTF { #1 }{
3939       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3940       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3941       \str_if_empty:NNTF \l_tmpa_str {
3942         \exp_args:Nx \cs_if_eq:NNTF {
3943           \tl_head:N \l_tmpa_tl
3944         } \stex_invoke_symbol:n {
3945           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3946         }{
3947           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3948         }
3949       } {
3950         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3951       }
3952     }{
3953       % argument is not a command name
3954       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3955       % \l_stex_all_symbols_seq
3956     }
3957   }
3958 }
3959
3960 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3961   \str_set:Nn \l_tmpa_str { #1 }
3962   \bool_set_false:N \l_tmpa_bool
3963   \bool_if:NF \l_tmpa_bool {
3964     \tl_set:Nn \l_tmpa_tl {
3965       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3966     }
3967     \str_set:Nn \l_tmpa_str { #1 }
3968     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3969     \seq_map_inline:Nn #2 {
3970       \str_set:Nn \l_tmpb_str { ##1 }
3971       \str_if_eq:eeT { \l_tmpa_str } {
3972         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3973       } {
3974         \seq_map_break:n {
3975           \tl_set:Nn \l_tmpa_tl {
3976             \str_set:Nn \l_stex_get_symbol_uri_str {
3977               ##1
3978             }
3979           }
3980         }
3981       }

```

```

3982     }
3983     \l_tmpa_tl
3984   }
3985 }
3986
3987 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3988   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3989     { \tl_tail:N \l_tmpa_tl }
3990   \tl_if_single:NTF \l_tmpa_tl {
3991     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3992       \exp_after:wN \str_set:Nn \exp_after:wN
3993         \l_stex_get_symbol_uri_str \l_tmpa_tl
3994       \__stex_copymodule_get_symbol_check:n { #1 }
3995     }{
3996       % TODO
3997       % tail is not a single group
3998     }
3999   }{
4000     % TODO
4001     % tail is not a single group
4002   }
4003 }
4004
4005 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
4006   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
4007     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
4008       :~\seq_use:Nn #1 {,~}
4009     }
4010   }
4011 }
4012
4013 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
4014   \stex_import_module_uri:nn { #1 } { #2 }
4015   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
4016   \stex_import_require_module:nnnn
4017     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
4018     { \l_stex_import_path_str } { \l_stex_import_name_str }
4019   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4020   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
4021   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
4022   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4023     \seq_map_inline:cn {c_stex_module_###_constants}{
4024       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
4025         ##1 ? ####1
4026       }
4027     }
4028   }
4029   \seq_clear:N \l_tmpa_seq
4030   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
4031     name      = \l_stex_current_copymodule_name_str ,
4032     module    = \l_stex_current_module_str ,
4033     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
4034     includes  = \l_tmpa_seq ,
4035     fields    = \l_tmpa_seq

```

```

4036 }
4037 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
4038   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
4039   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
4040 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
4041 \stex_if_smsmode:F {
4042   \begin{stex_annotate_env} {#4} {
4043     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4044   }
4045   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}}
4046 }
4047 %\bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
4048 %\bool_set_false:N \_stex_html_do_output_bool
4049 }
4050 \cs_new_protected:Nn \stex_copymodule_end:n {
4051   \def \l_tmpa_cs ##1 ##2 {#1}
4052   %\bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
4053   \tl_clear:N \l_tmpa_tl
4054   \tl_clear:N \l_tmpb_tl
4055   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4056   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
4057     \seq_map_inline:cn {c_stex_module_##1_constants}{
4058       \tl_clear:N \l_tmpc_tl
4059       \l_tmpa_cs{##1}{####1}
4060       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
4061         \stex_add_constant_to_current_module:n {\use:c{l__stex_copymodule_copymodule_##1?####1_
4062         \tl_put_right:Nx \l_tmpa_tl {
4063           \prop_set_from_keyval:cn {
4064             l_stex_symdecl\_l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
4065           }}{
4066             \exp_after:wN \prop_to_keyval:N \csname
4067               l_stex_symdecl\_l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
4068             \endcsname
4069           }
4070           \seq_clear:c {
4071             l_stex_symdecl_
4072             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
4073             _notations
4074           }
4075         }
4076         \tl_put_right:Nx \l_tmpc_tl {
4077           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
4078           \stex_if_smsmode:F{\stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_c
4079         }
4080         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
4081         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4082           \tl_put_right:Nx \l_tmpc_tl {
4083             \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymo
4084           }
4085           \tl_put_right:Nx \l_tmpa_tl {
4086             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4087               \stex_invoke_symbol:n {
4088                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
4089             }

```

```

4090     }
4091   }
4092 }
4093 }{
4094   \tl_put_right:Nx \l_tmpc_tl {
4095     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4096   }
4097   \stex_add_constant_to_current_module:n { \l_stex_current_copymodule_name_str / #####1
4098   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4099   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / #####1 }
4100   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
4101   \tl_put_right:Nx \l_tmpa_tl {
4102     \prop_set_from_keyval:cn {
4103       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
4104     }{
4105       \prop_to_keyval:N \l_tmpa_prop
4106     }
4107     \seq_clear:c {
4108       l_stex_symdecl_
4109       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
4110       _notations
4111     }
4112   }
4113   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4114   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4115     \tl_put_right:Nx \l_tmpc_tl {
4116       \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4117     }
4118     \tl_put_right:Nx \l_tmpa_tl {
4119       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4120         \stex_invoke_symbol:n {
4121           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / #####1
4122         }
4123       }
4124     }
4125   }
4126 }
4127 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4128   \tl_put_right:Nx \l_tmpc_tl {
4129     \stex_if_smsmode:F{
4130       $\stex_annotate_invisible:nnn{definiens}}{\exp_after:wN \exp_not:N\csname l__stex_copymodule_copymodule_##1?####1_def_tl\endcsname}
4131     }
4132   }
4133 }
4134 \tl_put_right:Nx \l_tmpb_tl {
4135   \stex_if_smsmode:TF{
4136     \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl}
4137   }{
4138     \stex_annotate:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl} }
4139   }
4140 }
4141 }
4142 }
4143 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq

```

```

4144 \tl_put_left:Nx \l_tmpa_tl {
4145   \prop_set_from_keyval:cn {
4146     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4147   }{
4148     \prop_to_keyval:N \l_stex_current_copymodule_prop
4149   }
4150 }
4151 \seq_gput_right:cx{c_stex_module_ \l_stex_current_module_str _copymodules}{
4152   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4153 }
4154 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4155 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
4156 \exp_args:Nx \stex_do_up_to_module:n {
4157   \exp_args:No \exp_not:n \l_tmpa_tl
4158 }
4159 \stex_debug:nn{copymodule}{output:\meaning \l_tmpb_tl}
4160 \l_tmpb_tl
4161 \stex_if_smsmode:F {
4162   \end{stex_annotate_env}
4163 }
4164 }
4165
4166 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4167   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4168   \stex_deactivate_macro:Nn \symdecl {module~environments}
4169   \stex_deactivate_macro:Nn \symdef {module~environments}
4170   \stex_deactivate_macro:Nn \notation {module~environments}
4171   \stex_reactivate_macro:N \assign
4172   \stex_reactivate_macro:N \renamedekl
4173   \stex_reactivate_macro:N \donotcopy
4174   \stex_smsmode_do:
4175 }{
4176   \stex_copymodule_end:n {}
4177 }
4178
4179 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4180   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4181   \stex_deactivate_macro:Nn \symdecl {module~environments}
4182   \stex_deactivate_macro:Nn \symdef {module~environments}
4183   \stex_deactivate_macro:Nn \notation {module~environments}
4184   \stex_reactivate_macro:N \assign
4185   \stex_reactivate_macro:N \renamedekl
4186   \stex_reactivate_macro:N \donotcopy
4187   \stex_smsmode_do:
4188 }{
4189   \stex_copymodule_end:n {
4190     \tl_if_exist:cF {
4191       l__stex_copymodule_copymodule_##1?##2_def_tl
4192     }{
4193       \str_if_eq:eeF {
4194         \prop_item:cn{
4195           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4196       }{ true }{
4197         \msg_error:nxxx{stex}{error/interpretmodule/noddefinens}{

```

```

4198         ##1?##2
4199     }\l_stex_current_copymodule_name_str}
4200 }
4201 }
4202 }
4203 }
4204
4205 \NewDocumentCommand \donotcopy { m }{
4206   \str_clear:N \l_stex_import_name_str
4207   \str_set:Nn \l_tmpa_str { #1 }
4208   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4209   \seq_map_inline:Nn \l_stex_all_modules_seq {
4210     \str_set:Nn \l_tmpb_str { ##1 }
4211     \str_if_eq:eeT { \l_tmpa_str } {
4212       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4213     } {
4214       \seq_map_break:n {
4215         \stex_if_do_html:T {
4216           \stex_if_smsmode:F {
4217             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4218               \stex_annotate:nnn{domain}{##1}{}}
4219           }
4220         }
4221       }
4222       \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4223     }
4224   }
4225   \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4226     \str_set:Nn \l_tmpb_str { #####1 }
4227     \str_if_eq:eeT { \l_tmpa_str } {
4228       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4229     } {
4230       \seq_map_break:n {\seq_map_break:n {
4231         \stex_if_do_html:T {
4232           \stex_if_smsmode:F {
4233             \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4234               \stex_annotate:nnn{domain}{
4235                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4236               }{}
4237             }
4238           }
4239         }
4240         \str_set:Nx \l_stex_import_name_str {
4241           \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4242         }
4243       }}
4244     }
4245   }
4246 }
4247 \str_if_empty:NTF \l_stex_import_name_str {
4248   % TODO throw error
4249 }{
4250   \stex_collect_imports:n {\l_stex_import_name_str }
4251   \seq_map_inline:Nn \l_stex_collect_imports_seq {

```

```

4252 \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4253 \seq_map_inline:cn {c_stex_module_##1_constants}{
4254 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4255 \bool_lazy_any:nT {
4256 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4257 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4258 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4259 }{
4260 % TODO throw error
4261 }
4262 }
4263 }
4264 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4265 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4266 \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4267 }
4268 \stex_smsmode_do:
4269 }
4270
4271 \NewDocumentCommand \assign { m m }{
4272 \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4273 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4274 \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4275 \stex_smsmode_do:
4276 }
4277
4278 \keys_define:nn { stex / renamedec1 } {
4279 name .str_set_x:N = \l_stex_renamedec1_name_str
4280 }
4281 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4282 \str_clear:N \l_stex_renamedec1_name_str
4283 \keys_set:nn { stex / renamedec1 } { #1 }
4284 }
4285
4286 \NewDocumentCommand \renamedec1 { 0{} m m }{
4287 \__stex_copymodule_renamedec1_args:n { #1 }
4288 \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4289 \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4290 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4291 \str_if_empty:NTF \l_stex_renamedec1_name_str {
4292 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4293 \l_stex_get_symbol_uri_str
4294 } }
4295 } {
4296 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4297 \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4298 \prop_set_eq:cc {l_stex_symdecl_
4299 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4300 _prop
4301 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4302 \seq_set_eq:cc {l_stex_symdecl_
4303 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4304 _notations
4305 }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}

```

```

4306 \prop_put:cnx {l_stex_symdecl_
4307 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4308 _prop
4309 }{ name }{ \l_stex_renameddecl_name_str }
4310 \prop_put:cnx {l_stex_symdecl_
4311 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4312 _prop
4313 }{ module }{ \l_stex_current_module_str }
4314 \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4315 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4316 }
4317 \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4318 \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4319 } }
4320 }
4321 \stex_smsmode_do:
4322 }
4323
4324 \stex_deactivate_macro:Nn \assign {copymodules}
4325 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4326 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4327
4328
4329 \seq_new:N \l_stex_implicit_morphisms_seq
4330 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4331 \stex_import_module_uri:nn { #1 } { #2 }
4332 \stex_debug:nn{implicits}{
4333 Implicit~morphism:~
4334 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4335 }
4336 \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4337 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4338 }{
4339 \msg_error:nnn{stex}{error/conflictingmodules}{
4340 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4341 }
4342 }
4343
4344 % TODO
4345
4346
4347
4348 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4349 \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4350 }
4351 }
4352

```

31.2 The feature environment

structural@feature

```

4353 <@@=stex_features>
4354

```



```

4355 \NewDocumentEnvironment{structural_feature_module}{m m m}{
4356   \stex_if_in_module:F {
4357     \msg_set:nnn{stex}{error/nomodule}{
4358       Structural~Feature~has~to~occur~in~a~module:\\
4359       Feature~#2~of~type~#1\\
4360       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4361     }
4362     \msg_error:nn{stex}{error/nomodule}
4363   }
4364
4365   \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str
4366
4367   \stex_module_setup:nn{meta=NONE}{#2 - #1}
4368
4369   \stex_if_smsmode:F {
4370     \begin{stex_annotate_env}{feature:#1 }{\l_tmpa_str ? #2 - #1}
4371     \stex_annotate_invisible:nnn{header}{}{ #3 }
4372   }
4373 }{
4374   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4375   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4376   \stex_debug:nn{features}{
4377     Feature: \l_stex_last_feature_str
4378   }
4379   \stex_if_smsmode:F {
4380     \end{stex_annotate_env}
4381   }
4382 }

```

31.3 Structure

structure

```

4383 <@@=stex_structures>
4384 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4385   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4386     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4387   }
4388   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4389   {#1}{#2}
4390 }
4391
4392 \keys_define:nn { stex / features / structure } {
4393   name .str_set_x:N = \l__stex_structures_name_str ,
4394 }
4395
4396 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4397   \str_clear:N \l__stex_structures_name_str
4398   \keys_set:nn { stex / features / structure } { #1 }
4399 }
4400
4401 \NewDocumentEnvironment{mathstructure}{m O{}}{
4402   \__stex_structures_structure_args:n { #2 }
4403   \str_if_empty:NT \l__stex_structures_name_str {

```

```

4404 \str_set:Nx \l__stex_structures_name_str { #1 }
4405 }
4406 \stex_suppress_html:n {
4407 \exp_args:Nx \stex_symdecl_do:nn {
4408 name = \l__stex_structures_name_str ,
4409 def = {\STEXsymbol{module-type}}{
4410 \_stex_term_math_oms:nnnn {
4411 \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4412 { ns } ?
4413 \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4414 { name } / \l__stex_structures_name_str - structure
4415 }{}{}{}
4416 }}
4417 }{ #1 }
4418 }
4419 \exp_args:Nnnx
4420 \begin{structural_feature_module}{ structure }
4421 { \l__stex_structures_name_str }{}
4422 \stex_smsmode_do:
4423 }{
4424 \end{structural_feature_module}
4425 \stex_reset_up_to_module:n \l_stex_last_feature_str
4426 \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4427 \seq_clear:N \l_tmpa_seq
4428 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4429 \seq_map_inline:cn{c_stex_module_##1_constants}{
4430 \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4431 }
4432 }
4433 \exp_args:Nnno
4434 \prop_gput:cn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4435 \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4436 \stex_add_structure_to_current_module:nn
4437 \l__stex_structures_name_str
4438 \l_stex_last_feature_str
4439 \exp_args:Nx
4440 \stex_add_to_current_module:n {
4441 \tl_set:cn { #1 }{
4442 \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4443 }
4444 }
4445 \exp_args:Nx
4446 \stex_do_up_to_module:n {
4447 \tl_set:cn { #1 }{
4448 \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4449 }
4450 }
4451 }
4452
4453 \cs_new:Nn \stex_invoke_structure:nn {
4454 \stex_invoke_symbol:n { #1?#2 }
4455 }
4456
4457 \cs_new_protected:Nn \stex_get_structure:n {

```

```

4458 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4459   \tl_set:Nn \l_tmpa_tl { #1 }
4460   \__stex_structures_get_from_cs:
4461 }{
4462   \cs_if_exist:cTF { #1 }{
4463     \cs_set_eq:Nc \l_tmpa_cs { #1 }
4464     \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4465     \str_if_empty:NTF \l_tmpa_str {
4466       \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4467         \__stex_structures_get_from_cs:
4468       }{
4469         \__stex_structures_get_from_string:n { #1 }
4470       }
4471     }{
4472       \__stex_structures_get_from_string:n { #1 }
4473     }
4474   }{
4475     \__stex_structures_get_from_string:n { #1 }
4476   }
4477 }
4478 }
4479
4480 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4481   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4482   { \tl_tail:N \l_tmpa_tl }
4483   \str_set:Nx \l_tmpa_str {
4484     \exp_after:wN \use_i:nn \l_tmpa_tl
4485   }
4486   \str_set:Nx \l_tmpb_str {
4487     \exp_after:wN \use_ii:nn \l_tmpa_tl
4488   }
4489   \str_set:Nx \l_stex_get_structure_str {
4490     \l_tmpa_str ? \l_tmpb_str
4491   }
4492   \str_set:Nx \l_stex_get_structure_module_str {
4493     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4494   }
4495 }
4496
4497 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4498   \tl_set:Nn \l_tmpa_tl {
4499     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4500   }
4501   \str_set:Nn \l_tmpa_str { #1 }
4502   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4503
4504   \seq_map_inline:Nn \l_stex_all_modules_seq {
4505     \prop_if_exist:cT {c_stex_module_##1_structures} {
4506       \prop_map_inline:cn {c_stex_module_##1_structures} {
4507         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4508           \prop_map_break:n{\seq_map_break:n{
4509             \tl_set:Nn \l_tmpa_tl {
4510               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4511               \str_set:Nn \l_stex_get_structure_module_str {####2}

```

```

4512     }
4513   }}
4514 }
4515 }
4516 }
4517 }
4518 \l_tmpa_tl
4519 }

```

\instantiate

```

4520
4521 \keys_define:nn { stex / instantiate } {
4522   name .str_set_x:N = \l__stex_structures_name_str
4523 }
4524 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4525   \str_clear:N \l__stex_structures_name_str
4526   \keys_set:nn { stex / instantiate } { #1 }
4527 }
4528
4529 \NewDocumentCommand \instantiate {m O{} m m m}{
4530   \begin{group}
4531     \stex_get_structure:n {#4}
4532     \__stex_structures_instantiate_args:n { #2 }
4533     \str_if_empty:NT \l__stex_structures_name_str {
4534       \str_set:Nn \l__stex_structures_name_str { #1 }
4535     }
4536     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4537     \seq_clear:N \l__stex_structures_fields_seq
4538     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4539     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4540       \seq_map_inline:cn {c_stex_module_##1_constants}{
4541         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4542       }
4543     }
4544
4545     \tl_if_empty:nF{#3}{
4546       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4547       \prop_clear:N \l_tmpa_prop
4548       \seq_map_inline:Nn \l_tmpa_seq {
4549         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4550         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4551           \msg_error:nnn{stex}{error/keyval}{##1}
4552         }
4553         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4554         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4555         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4556         \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4557         \exp_args:Nxx \str_if_eq:nnF
4558           {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4559           {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4560           \msg_error:nnxxx{stex}{error/incompatible}
4561           {\l__stex_structures_dom_str}
4562           {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4563           {\l_stex_get_symbol_uri_str}

```

```

4564         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4565     }
4566     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4567 }
4568 }
4569
4570
4571
4572
4573 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4574     \str_set:Nx \l_tmpa_str {field:\l__stex_structures_name_str . \prop_item:cn {l_stex_sy
4575     \stex_debug:nn{instantiate}{Field~\l_tmpa_str :~##1}
4576
4577     \exp_args:Nx \stex_do_up_to_module:n {
4578         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4579             name = \l_tmpa_str ,
4580             args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4581             arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4582             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4583         }
4584         \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notations}
4585         \stex_add_constant_to_current_module:n {\l_tmpa_str}
4586     }
4587     \exp_args:Nx \stex_add_to_current_module:n {
4588         \prop_set_from_keyval:cn { l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _p
4589             name = \l_tmpa_str ,
4590             args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4591             arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4592             assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4593         }
4594         \seq_clear:c {l_stex_symdecl_ \l_stex_current_module_str?\l_tmpa_str _notations}
4595         \stex_add_constant_to_current_module:n {\l_tmpa_str}
4596     }
4597
4598
4599
4600 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4601     \stex_find_notation:nn{##1}{}
4602     \exp_args:Nx\stex_do_up_to_module:n {
4603         \seq_put_right:cn {l_stex_symdecl_\l_stex_current_module_str?\l_tmpa_str _notation
4604     }
4605     \exp_args:Nx\stex_add_to_current_module:n {
4606         \seq_put_right:cn {l_stex_symdecl_f\l_stex_current_module_str?\l_tmpa_str _notatio
4607     }
4608
4609     \stex_copy_control_sequence:ccN
4610         {stex_notation_\l_stex_current_module_str?\l_tmpa_str _cs}
4611         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4612         \l_tmpa_tl
4613     \exp_args:No \stex_do_up_to_module:n \l_tmpa_tl
4614     \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4615
4616
4617     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{

```

```

4618         \tl_set_eq:Nc \l_tmpa_cs {stex_op_notation_##1\c_hash_str \l_stex_notation_variant
4619         \tl_set:Nx \l_tmpa_tl {
4620             \tl_set:cn
4621                 {stex_op_notation_\l_stex_current_module_str?\l_tmpa_str_cs}
4622                 { \exp_args:No \exp_not:n \l_tmpa_cs}
4623         }
4624     }
4625
4626 }
4627
4628 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\l_stex_cur
4629 }
4630
4631
4632 %\seq_if_empty:NF \l__stex_structures_fields_seq {
4633 % \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields
4634 %}
4635 \exp_args:Nx
4636 \stex_add_to_current_module:n {
4637     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4638     domain = \l_stex_get_structure_module_str ,
4639     \prop_to_keyval:N \l_tmpa_prop
4640 }
4641 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4642 }
4643 \exp_args:Nx
4644 \stex_do_up_to_module:n {
4645     \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4646     domain = \l_stex_get_structure_module_str ,
4647     \prop_to_keyval:N \l_tmpa_prop
4648 }
4649 \tl_set:cn{ #1 }{\stex_invoke_instance:n{\l_stex_current_module_str?\l__stex_structur
4650 }
4651 \stex_debug:nn{instantiate}{
4652     Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4653     \prop_to_keyval:N \l_tmpa_prop
4654 }
4655 \exp_args:Nxx \stex_symdecl_do:nn {
4656     type={\STEXsymbol{module-type}}{
4657         \_stex_term_math_oms:nnnn {
4658             \l_stex_get_structure_module_str
4659         }{}{0}{}
4660     }}
4661 }{\l__stex_structures_name_str}
4662 \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4663 \endgroup
4664 \stex_smsmode_do:\ignorespacesandpars
4665 }
4666
4667 \cs_new_protected:Nn \stex_symbol_or_var:n {
4668     \cs_if_exist:cTF{#1}{
4669         \cs_set_eq:Nc \l_tmpa_tl { #1 }
4670         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4671         \str_if_empty:NTF \l_tmpa_str {

```

```

4672 \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4673 \stex_invoke_variable:n {
4674   \bool_set_true:N \l_stex_symbol_or_var_bool
4675   \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4676   \str_set:Nx \l_stex_get_symbol_uri_str {
4677     \exp_after:wN \use:n \l_tmpa_tl
4678   }
4679   }{
4680     \bool_set_false:N \l_stex_symbol_or_var_bool
4681     \stex_get_symbol:n{#1}
4682   }
4683   }{
4684     \__stex_structures_symbolorvar_from_string:n{ #1 }
4685   }
4686   }{
4687     \__stex_structures_symbolorvar_from_string:n{ #1 }
4688   }
4689 }
4690
4691 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4692   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4693     \bool_set_true:N \l_stex_symbol_or_var_bool
4694     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4695   }{
4696     \bool_set_false:N \l_stex_symbol_or_var_bool
4697     \stex_get_symbol:n{#1}
4698   }
4699 }
4700
4701 \keys_define:nn { stex / varinstantiate } {
4702   name .str_set_x:N = \l__stex_structures_name_str,
4703   bind .choices:nn =
4704     {forall,exists}
4705     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4706 }
4707
4708 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4709   \str_clear:N \l__stex_structures_name_str
4710   \str_clear:N \l__stex_structures_bind_str
4711   \keys_set:nn { stex / varinstantiate } { #1 }
4712 }
4713
4714 \NewDocumentCommand \varinstantiate {m O{}} m m m m){
4715   \begin{group}
4716     \stex_get_structure:n {#4}
4717     \__stex_structures_varinstantiate_args:n { #2 }
4718     \str_if_empty:NT \l__stex_structures_name_str {
4719       \str_set:Nn \l__stex_structures_name_str { #1 }
4720     }
4721     \stex_if_do_html:TF{
4722       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4723     }{\use:n}
4724     {
4725       \stex_if_do_html:T{

```

```

4726     \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{}}
4727 }
4728 \seq_clear:N \l__stex_structures_fields_seq
4729 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4730 \seq_map_inline:Nn \l_stex_collect_imports_seq {
4731     \seq_map_inline:cn {c_stex_module_##1_constants}{
4732         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4733     }
4734 }
4735 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4736 \prop_clear:N \l_tmpa_prop
4737 \tl_if_empty:nF {#3} {
4738     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4739     \seq_map_inline:Nn \l_tmpa_seq {
4740         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4741         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4742             \msg_error:nnn{stex}{error/keyval}{##1}
4743         }
4744         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_fields_seq
4745         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4746         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
4747         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4748         \stex_if_do_html:T{
4749             \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}{##1}
4750         }
4751         \bool_if:NTF \l_stex_symbol_or_var_bool {
4752             \exp_args:Nxx \str_if_eq:nnF
4753                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4754                 {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4755                 \msg_error:nnxxx{stex}{error/incompatible}
4756                 {\l__stex_structures_dom_str}
4757                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4758                 {\l_stex_get_symbol_uri_str}
4759                 {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4760             }
4761             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {##1}}
4762         }{
4763             \exp_args:Nxx \str_if_eq:nnF
4764                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4765                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4766                 \msg_error:nnxxx{stex}{error/incompatible}
4767                 {\l__stex_structures_dom_str}
4768                 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4769                 {\l_stex_get_symbol_uri_str}
4770                 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4771             }
4772             \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {##1}}
4773         }
4774     }
4775 }
4776 \tl_gclear:N \g__stex_structures_aftergroup_tl
4777 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4778     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str} {##1}}
4779     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}

```



```

4780 \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4781 \stex_find_notation:nn{##1}{}
4782 \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str_cs}
4783 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4784 \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_
4785 \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4786 \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4787 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4788 \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_struct
4789 }
4790 }
4791
4792 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4793 \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4794 name = \l_tmpa_str ,
4795 args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4796 arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4797 assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4798 }
4799 \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4800 {g__stex_structures_tmpa_\l_tmpa_str_cs}
4801 \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4802 {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4803 }
4804 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4805 }
4806 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4807 \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4808 domain = \l_stex_get_structure_module_str ,
4809 \prop_to_keyval:N \l_tmpa_prop
4810 }
4811 \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4812 \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4813 \exp_args:Nnx \exp_not:N \use:nn {
4814 \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4815 \stex_term_omv:nn {var://\l__stex_structures_name_str}{
4816 \exp_not:n{
4817 \varcomp{#5}
4818 }
4819 }
4820 }{
4821 \exp_not:n{\stex_reset:N \l_stex_current_symbol_str}
4822 }
4823 }
4824 }
4825 }
4826 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{g__stex_structures_a
4827 \aftergroup\g__stex_structures_aftergroup_tl
4828 \endgroup
4829 \stex_smsmode_do:\ignorespacesandpars
4830 }
4831
4832 \cs_new_protected:Nn \stex_invoke_instance:n {
4833 \peek_charcode_remove:NTF ! {

```

```

4834     \stex_invoke_symbol:n{#1}
4835   }{
4836     \_stex_invoke_instance:nn {#1}
4837   }
4838 }
4839
4840
4841 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4842   \peek_charcode_remove:NTF ! {
4843     \exp_args:Nnx \use:nn {
4844       \def\comp{\_varcomp}
4845       \use:c{l_stex_varinstance_#1_op_tl}
4846     }{
4847       \_stex_reset:N \comp
4848     }
4849   }{
4850     \_stex_invoke_varinstance:nn {#1}
4851   }
4852 }
4853
4854 \cs_new_protected:Nn \_stex_invoke_instance:nn {
4855   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4856     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4857   }{
4858     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4859     \msg_error:nnxxx{stex}{error/unknownfield}{#2}{#1}{
4860       \prop_to_keyval:N \l_tmpa_prop
4861     }
4862   }
4863 }
4864
4865 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4866   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4867     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4868     \l_tmpa_tl
4869   }{
4870     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
4871     }
4872   }

```

(End definition for \instantiate. This function is documented on page 31.)

\stex_invoke_structure:nnn

```

4873 % #1: URI of the instance
4874 % #2: URI of the instantiated module
4875 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4876   \tl_if_empty:nTF{ #3 }{
4877     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4878       c_stex_feature_ #2 _prop
4879     }
4880     \tl_clear:N \l_tmpa_tl
4881     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4882     \seq_map_inline:Nn \l_tmpa_seq {
4883       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }

```

```

4884 \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4885 \cs_if_exist:cT {
4886   stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4887 }{
4888   \tl_if_empty:NF \l_tmpa_tl {
4889     \tl_put_right:Nn \l_tmpa_tl {,}
4890   }
4891   \tl_put_right:Nx \l_tmpa_tl {
4892     \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4893   }
4894 }
4895 }
4896 \exp_args:No \mathstruct \l_tmpa_tl
4897 }{
4898   \stex_invoke_symbol:n{#1/#3}
4899 }
4900 }

```

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

```

4901 \endpackage

```

Chapter 32

STEX -Statements Implementation

```
4902 <*package>
4903
4904 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4905
4906 <@@=stex_statements>
    Warnings and error messages
4907

\titleemph
4908 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
4909 \keys_define:nn {stex / definiendum }{
4910   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4911   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4912   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4913   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4914 }
4915 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4916   \str_clear:N \l__stex_statements_definiendum_root_str
4917   \tl_clear:N \l__stex_statements_definiendum_post_tl
4918   \str_clear:N \l__stex_statements_definiendum_gfa_str
4919   \keys_set:nn { stex / definiendum }{ #1 }
4920 }
4921 \NewDocumentCommand \definiendum { O{} m m } {
4922   \__stex_statements_definiendum_args:n { #1 }
4923   \stex_get_symbol:n { #2 }
4924   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4925   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4926     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4927     \tl_set:Nn \l_tmpa_tl { #3 }
4928   } {
4929     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4930     \tl_set:Nn \l_tmpa_tl {
4931       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4932     }
4933   }
4934 } {
4935   \tl_set:Nn \l_tmpa_tl { #3 }
4936 }
4937
4938 % TODO root
4939 \rustex_if:TF {
4940   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4941 } {
4942   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4943 }
4944 }
4945 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

(End definition for definiendum. This function is documented on page 40.)

definame

```

4946
4947 \NewDocumentCommand \definame { 0{ } m } {
4948   \__stex_statements_definiendum_args:n { #1 }
4949   % TODO: root
4950   \stex_get_symbol:n { #2 }
4951   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4952   \str_set:Nx \l_tmpa_str {
4953     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4954   }
4955   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4956   \rustex_if:TF {
4957     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4958       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4959     }
4960   } {
4961     \exp_args:Nnx \defemph@uri {
4962       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4963     } { \l_stex_get_symbol_uri_str }
4964   }
4965 }
4966 \stex_deactivate_macro:Nn \definame {definition~environments}
4967
4968 \NewDocumentCommand \Definame { 0{ } m } {
4969   \__stex_statements_definiendum_args:n { #1 }
4970   \stex_get_symbol:n { #2 }
4971   \str_set:Nx \l_tmpa_str {
4972     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4973   }
4974   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4975   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4976   \rustex_if:TF {

```

```

4977 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4978 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4979 }
4980 } {
4981 \exp_args:Nnx \defemph@uri {
4982 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4983 } { \l_stex_get_symbol_uri_str }
4984 }
4985 }
4986 \stex_deactivate_macro:Nn \Definame {definition~environments}
4987
4988 \NewDocumentCommand \premise { m }{
4989 \stex_annotate:nnn{ premise }{}{ #1 }
4990 }
4991 \NewDocumentCommand \conclusion { m }{
4992 \stex_annotate:nnn{ conclusion }{}{ #1 }
4993 }
4994 \NewDocumentCommand \definiens { O{} m }{
4995 \str_clear:N \l_stex_get_symbol_uri_str
4996 \tl_if_empty:nF {#1} {
4997 \stex_get_symbol:n { #1 }
4998 }
4999 \str_if_empty:NT \l_stex_get_symbol_uri_str {
5000 \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
5001 \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
5002 }{
5003 % TODO throw error
5004 }
5005 }
5006 \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}
5007 {\l_stex_current_module_str}{
5008 \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
5009 {true}}{
5010 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5011 \exp_args:Nx \stex_add_to_current_module:n {
5012 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
5013 }
5014 }
5015 }
5016 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
5017 }
5018
5019 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
5020 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
5021 \stex_deactivate_macro:Nn \definiens {definition~environments}
5022

```

(End definition for `definame`. This function is documented on page 40.)

sdefinition

```

5023
5024 \keys_define:nn {stex / sdefinition }{
5025 type .str_set_x:N = \sdefinitiontype,
5026 id .str_set_x:N = \sdefinitionid,

```

```

5027 name .str_set_x:N = \sdefinitionname,
5028 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5029 title .tl_set:N = \sdefinitiontitle
5030 }
5031 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
5032 \str_clear:N \sdefinitiontype
5033 \str_clear:N \sdefinitionid
5034 \str_clear:N \sdefinitionname
5035 \clist_clear:N \l__stex_statements_sdefinition_for_clist
5036 \tl_clear:N \sdefinitiontitle
5037 \keys_set:nn { stex / sdefinition }{ #1 }
5038 }
5039
5040 \NewDocumentEnvironment{sdefinition}{0{}}{
5041 \__stex_statements_sdefinition_args:n{ #1 }
5042 \stex_reactivate_macro:N \definiendum
5043 \stex_reactivate_macro:N \definame
5044 \stex_reactivate_macro:N \Definame
5045 \stex_reactivate_macro:N \premise
5046 \stex_reactivate_macro:N \definiens
5047 \stex_if_smsmode:F{
5048 \seq_clear:N \l_tmpa_seq
5049 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5050 \tl_if_empty:NF{ ##1 }{
5051 \stex_get_symbol:n { ##1 }
5052 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5053 \l_stex_get_symbol_uri_str
5054 }
5055 }
5056 }
5057 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5058 \exp_args:Nnnx
5059 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
5060 \str_if_empty:NF \sdefinitiontype {
5061 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5062 }
5063 \str_if_empty:NF \sdefinitionname {
5064 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5065 }
5066 \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5067 \tl_clear:N \l_tmpa_tl
5068 \clist_map_inline:Nn \l_tmpa_clist {
5069 \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
5070 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
5071 }
5072 }
5073 \tl_if_empty:NTF \l_tmpa_tl {
5074 \__stex_statements_sdefinition_start:
5075 }{
5076 \l_tmpa_tl
5077 }
5078 }
5079 \stex_ref_new_doc_target:n \sdefinitionid
5080 \stex_smsmode_do:

```

```

5081 }{
5082   \stex_suppress_html:n {
5083     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5084   }
5085   \stex_if_smsmode:F {
5086     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
5087     \tl_clear:N \l_tmpa_tl
5088     \clist_map_inline:Nn \l_tmpa_clist {
5089       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
5090         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
5091       }
5092     }
5093     \tl_if_empty:NTF \l_tmpa_tl {
5094       \__stex_statements_sdefinition_end:
5095     }{
5096       \l_tmpa_tl
5097     }
5098     \end{stex_annotate_env}
5099   }
5100 }

```

\stexpatchdefinition

```

5101 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
5102   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
5103     ~(\sdefinitiontitle)
5104   }~}
5105 }
5106 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
5107
5108 \newcommand\stexpatchdefinition[3] [] {
5109   \str_set:Nx \l_tmpa_str{ #1 }
5110   \str_if_empty:NTF \l_tmpa_str {
5111     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
5112     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
5113   }{
5114     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
5115     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
5116   }
5117 }

```

(End definition for \stexpatchdefinition. This function is documented on page [42](#).)

\inlinedef inline:

```

5118 \keys_define:nn {stex / inlinedef }{
5119   type      .str_set_x:N = \sdefinitiontype,
5120   id        .str_set_x:N = \sdefinitionid,
5121   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
5122   name      .str_set_x:N = \sdefinitionname
5123 }
5124 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
5125   \str_clear:N \sdefinitiontype
5126   \str_clear:N \sdefinitionid
5127   \str_clear:N \sdefinitionname
5128   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```



```

5129 \keys_set:nn { stex / inlinedef }{ #1 }
5130 }
5131 \NewDocumentCommand \inlinedef { 0{} m } {
5132   \beginngroup
5133   \__stex_statements_inlinedef_args:n{ #1 }
5134   \stex_reactivate_macro:N \definiendum
5135   \stex_reactivate_macro:N \definame
5136   \stex_reactivate_macro:N \Definame
5137   \stex_reactivate_macro:N \premise
5138   \stex_reactivate_macro:N \definiens
5139   \stex_ref_new_doc_target:n \sdefinitionid
5140   \stex_if_smsmode:TF{\stex_suppress_html:n {
5141     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5142   }}{
5143     \seq_clear:N \l_tmpa_seq
5144     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5145       \tl_if_empty:NF{ ##1 }{
5146         \stex_get_symbol:n { ##1 }
5147         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5148           \l_stex_get_symbol_uri_str
5149         }
5150       }
5151     }
5152     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5153     \exp_args:Nnx
5154     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5155       \str_if_empty:NF \sdefinitiontype {
5156         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5157       }
5158       #2
5159       \str_if_empty:NF \sdefinitionname {
5160         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5161         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5162       }
5163     }
5164   }
5165   \endgroup
5166   \stex_smsmode_do:
5167 }

```

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

32.2 Assertions

sassertion

```

5168
5169 \keys_define:nn {stex / sassertion }{
5170   type      .str_set_x:N = \sassertiontype,
5171   id        .str_set_x:N = \sassertionid,
5172   title     .tl_set:N    = \sassertiontitle ,
5173   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5174   name      .str_set_x:N = \sassertionname
5175 }

```

```

5176 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5177   \str_clear:N \sassertiontype
5178   \str_clear:N \sassertionid
5179   \str_clear:N \sassertionname
5180   \clist_clear:N \l__stex_statements_sassertion_for_clist
5181   \tl_clear:N \sassertiontitle
5182   \keys_set:nn { stex / sassertion }{ #1 }
5183 }
5184
5185 %\tl_new:N \g__stex_statements_aftergroup_tl
5186
5187 \NewDocumentEnvironment{sassertion}{0{}}{
5188   \__stex_statements_sassertion_args:n{ #1 }
5189   \stex_reactivate_macro:N \premise
5190   \stex_reactivate_macro:N \conclusion
5191   \stex_if_smsmode:F {
5192     \seq_clear:N \l_tmpa_seq
5193     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5194       \tl_if_empty:NF{ ##1 }{
5195         \stex_get_symbol:n { ##1 }
5196         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5197           \l_stex_get_symbol_uri_str
5198         }
5199       }
5200     }
5201     \exp_args:Nnnx
5202     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5203     \str_if_empty:NF \sassertiontype {
5204       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5205     }
5206     \str_if_empty:NF \sassertionname {
5207       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5208     }
5209     \clist_set:Nn \l_tmpa_clist \sassertiontype
5210     \tl_clear:N \l_tmpa_tl
5211     \clist_map_inline:Nn \l_tmpa_clist {
5212       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5213         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5214       }
5215     }
5216     \tl_if_empty:NTF \l_tmpa_tl {
5217       \__stex_statements_sassertion_start:
5218     }{
5219       \l_tmpa_tl
5220     }
5221   }
5222   \str_if_empty:NTF \sassertionid {
5223     \str_if_empty:NF \sassertionname {
5224       \stex_ref_new_doc_target:n { }
5225     }
5226   } {
5227     \stex_ref_new_doc_target:n \sassertionid
5228   }
5229   \stex_smsmode_do:

```

```

5230 }{
5231   \str_if_empty:NF \sassertionname {
5232     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5233     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5234   }
5235   \stex_if_smsmode:F {
5236     \clist_set:Nn \l_tmpa_clist \sassertiontype
5237     \tl_clear:N \l_tmpa_tl
5238     \clist_map_inline:Nn \l_tmpa_clist {
5239       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5240         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5241       }
5242     }
5243     \tl_if_empty:NTF \l_tmpa_tl {
5244       \__stex_statements_sassertion_end:
5245     }{
5246       \l_tmpa_tl
5247     }
5248     \end{stex_annotate_env}
5249   }
5250 }

```

\stexpatchassertion

```

5251
5252 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5253   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5254     (\sassertiontitle)
5255   }~}
5256 }
5257 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5258
5259 \newcommand\stexpatchassertion[3] [] {
5260   \str_set:Nx \l_tmpa_str{ #1 }
5261   \str_if_empty:NTF \l_tmpa_str {
5262     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5263     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5264   }{
5265     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5266     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5267   }
5268 }

```

(End definition for \stexpatchassertion. This function is documented on page [42](#).)

\inlineass inline:

```

5269 \keys_define:nn {stex / inlineass }{
5270   type      .str_set_x:N = \sassertiontype,
5271   id        .str_set_x:N = \sassertionid,
5272   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5273   name      .str_set_x:N = \sassertionname
5274 }
5275 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5276   \str_clear:N \sassertiontype
5277   \str_clear:N \sassertionid

```

```

5278 \str_clear:N \sassertionname
5279 \clist_clear:N \l__stex_statements_sassertion_for_clist
5280 \keys_set:nn { stex / inlineass }{ #1 }
5281 }
5282 \NewDocumentCommand \inlineass { 0{} m } {
5283 \beginngroup
5284 \stex_reactivate_macro:N \premise
5285 \stex_reactivate_macro:N \conclusion
5286 \__stex_statements_inlineass_args:n{ #1 }
5287 \str_if_empty:NTF \sassertionid {
5288 \str_if_empty:NF \sassertionname {
5289 \stex_ref_new_doc_target:n {}
5290 }
5291 } {
5292 \stex_ref_new_doc_target:n \sassertionid
5293 }
5294
5295 \stex_if_smsmode:TF{
5296 \str_if_empty:NF \sassertionname {
5297 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5298 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5299 }
5300 }{
5301 \seq_clear:N \l_tmpa_seq
5302 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5303 \tl_if_empty:NF{ ##1 }{
5304 \stex_get_symbol:n { ##1 }
5305 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5306 \l_stex_get_symbol_uri_str
5307 }
5308 }
5309 }
5310 \exp_args:Nnx
5311 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5312 \str_if_empty:NF \sassertiontype {
5313 \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5314 }
5315 #2
5316 \str_if_empty:NF \sassertionname {
5317 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5318 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5319 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5320 }
5321 }
5322 }
5323 \endgroup
5324 \stex_smsmode_do:
5325 }

```

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

32.3 Examples

sexample

```

5326 \keys_define:nn {stex / sexample }{
5327   type      .str_set_x:N = \exampletype,
5328   id        .str_set_x:N = \sexampleid,
5329   title     .tl_set:N     = \sexampletitle,
5330   name      .str_set_x:N = \sexamplename ,
5331   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5332 }
5333 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5334   \str_clear:N \sexampletype
5335   \str_clear:N \sexampleid
5336   \str_clear:N \sexamplename
5337   \tl_clear:N \sexampletitle
5338   \clist_clear:N \l__stex_statements_sexample_for_clist
5339   \keys_set:nn { stex / sexample }{ #1 }
5340 }
5341
5342
5343 \NewDocumentEnvironment{sexample}{0{}}{
5344   \__stex_statements_sexample_args:n{ #1 }
5345   \stex_reactivate_macro:N \premise
5346   \stex_reactivate_macro:N \conclusion
5347   \stex_if_smsmode:F {
5348     \seq_clear:N \l_tmpa_seq
5349     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5350       \tl_if_empty:NF{ ##1 }{
5351         \stex_get_symbol:n { ##1 }
5352         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5353           \l_stex_get_symbol_uri_str
5354         }
5355       }
5356     }
5357     \exp_args:Nnnx
5358     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5359     \str_if_empty:NF \sexampletype {
5360       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5361     }
5362     \str_if_empty:NF \sexamplename {
5363       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5364     }
5365     \clist_set:Nn \l_tmpa_clist \sexampletype
5366     \tl_clear:N \l_tmpa_tl
5367     \clist_map_inline:Nn \l_tmpa_clist {
5368       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5369         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5370       }
5371     }
5372     \tl_if_empty:NTF \l_tmpa_tl {
5373       \__stex_statements_sexample_start:
5374     }{
5375       \l_tmpa_tl
5376     }

```

```

5377 }
5378 \str_if_empty:NF \sexampleid {
5379   \stex_ref_new_doc_target:n \sexampleid
5380 }
5381 \stex_smsmode_do:
5382 }{
5383   \str_if_empty:NF \sexamplename {
5384     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5385   }
5386   \stex_if_smsmode:F {
5387     \clist_set:Nn \l_tmpa_clist \sexamplotype
5388     \tl_clear:N \l_tmpa_tl
5389     \clist_map_inline:Nn \l_tmpa_clist {
5390       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5391         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5392       }
5393     }
5394     \tl_if_empty:NTF \l_tmpa_tl {
5395       \__stex_statements_sexample_end:
5396     }{
5397       \l_tmpa_tl
5398     }
5399     \end{stex_annotate_env}
5400   }
5401 }

```

\stexpatchexample

```

5402
5403 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5404   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5405     (\sexamplotype)
5406   }~}
5407 }
5408 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5409
5410 \newcommand\stexpatchexample[3] [] {
5411   \str_set:Nx \l_tmpa_str{ #1 }
5412   \str_if_empty:NTF \l_tmpa_str {
5413     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5414     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5415   }{
5416     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5417     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5418   }
5419 }

```

(End definition for \stexpatchexample. This function is documented on page 42.)

\inlineex inline:

```

5420 \keys_define:nn {stex / inlineex }{
5421   type .str_set_x:N = \sexamplotype,
5422   id .str_set_x:N = \sexampleid,
5423   for .clist_set:N = \l__stex_statements_sexample_for_clist ,
5424   name .str_set_x:N = \sexamplename

```

```

5425 }
5426 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5427   \str_clear:N \sexamplotype
5428   \str_clear:N \sexampleid
5429   \str_clear:N \sexamplename
5430   \clist_clear:N \l__stex_statements_sexample_for_clist
5431   \keys_set:nn { stex / inlineex }{ #1 }
5432 }
5433 \NewDocumentCommand \inlineex { 0{ } m } {
5434   \beginngroup
5435   \stex_reactivate_macro:N \premise
5436   \stex_reactivate_macro:N \conclusion
5437   \__stex_statements_inlineex_args:n{ #1 }
5438   \str_if_empty:NF \sexampleid {
5439     \stex_ref_new_doc_target:n \sexampleid
5440   }
5441   \stex_if_smsmode:TF{
5442     \str_if_empty:NF \sexamplename {
5443       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5444     }
5445   }{
5446     \seq_clear:N \l_tmpa_seq
5447     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5448       \tl_if_empty:nF{ ##1 }{
5449         \stex_get_symbol:n { ##1 }
5450         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5451           \l_stex_get_symbol_uri_str
5452         }
5453       }
5454     }
5455     \exp_args:Nnx
5456     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5457       \str_if_empty:NF \sexamplotype {
5458         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5459       }
5460       #2
5461       \str_if_empty:NF \sexamplename {
5462         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5463         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5464       }
5465     }
5466   }
5467   \endgroup
5468   \stex_smsmode_do:
5469 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5470 \keys_define:nn { stex / sparagraph } {
5471   id .str_set_x:N = \sparagraphid ,

```

```

5472 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5473 type .str_set_x:N = \sparagraphtype ,
5474 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5475 from .tl_set:N = \sparagraphfrom ,
5476 to .tl_set:N = \sparagraphto ,
5477 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5478 name .str_set:N = \sparagraphname
5479 }
5480
5481 \cs_new_protected:Nn \stex_sparagraph_args:n {
5482 \tl_clear:N \l_stex_sparagraph_title_tl
5483 \tl_clear:N \sparagraphfrom
5484 \tl_clear:N \sparagraphto
5485 \tl_clear:N \l_stex_sparagraph_start_tl
5486 \str_clear:N \sparagraphid
5487 \str_clear:N \sparagraphtype
5488 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5489 \str_clear:N \sparagraphname
5490 \keys_set:nn { stex / sparagraph }{ #1 }
5491 }
5492 \newif\if@in@omtext\@in@omtextfalse
5493
5494 \NewDocumentEnvironment {sparagraph} { 0{} } {
5495 \stex_sparagraph_args:n { #1 }
5496 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5497 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5498 }{
5499 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5500 }
5501 \@in@omtexttrue
5502 \stex_if_smsmode:F {
5503 \seq_clear:N \l_tmpa_seq
5504 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5505 \tl_if_empty:NF{ ##1 }{
5506 \stex_get_symbol:n { ##1 }
5507 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5508 \l_stex_get_symbol_uri_str
5509 }
5510 }
5511 }
5512 \exp_args:Nnnx
5513 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5514 \str_if_empty:NF \sparagraphtype {
5515 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5516 }
5517 \str_if_empty:NF \sparagraphfrom {
5518 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5519 }
5520 \str_if_empty:NF \sparagraphto {
5521 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5522 }
5523 \str_if_empty:NF \sparagraphname {
5524 \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5525 }

```



```

5526 \clist_set:No \l_tmpa_clist \sparagraphtype
5527 \tl_clear:N \l_tmpa_tl
5528 \clist_map_inline:Nn \sparagraphtype {
5529   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5530     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5531   }
5532 }
5533 \tl_if_empty:NTF \l_tmpa_tl {
5534   \__stex_statements_sparagraph_start:
5535 }{
5536   \l_tmpa_tl
5537 }
5538 }
5539 \clist_set:No \l_tmpa_clist \sparagraphtype
5540 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}
5541 {
5542   \stex_reactivate_macro:N \definiendum
5543   \stex_reactivate_macro:N \definame
5544   \stex_reactivate_macro:N \Definame
5545   \stex_reactivate_macro:N \premise
5546   \stex_reactivate_macro:N \definens
5547 }
5548 \str_if_empty:NTF \sparagraphid {
5549   \str_if_empty:NTF \sparagraphname {
5550     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5551       \stex_ref_new_doc_target:n {}
5552     }
5553   } {
5554     \stex_ref_new_doc_target:n {}
5555   }
5556 } {
5557   \stex_ref_new_doc_target:n \sparagraphid
5558 }
5559 \exp_args:NNx
5560 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5561   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5562     \tl_if_empty:nF{ ##1 }{
5563       \stex_get_symbol:n { ##1 }
5564       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5565     }
5566   }
5567 }
5568 \stex_smsmode_do:
5569 \ignorespacesandpars
5570 }{
5571   \str_if_empty:NF \sparagraphname {
5572     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5573     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5574   }
5575   \stex_if_smsmode:F {
5576     \clist_set:No \l_tmpa_clist \sparagraphtype
5577     \tl_clear:N \l_tmpa_tl
5578     \clist_map_inline:Nn \l_tmpa_clist {
5579       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{

```

```

5580     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5581   }
5582 }
5583 \tl_if_empty:NTF \l_tmpa_tl {
5584   \__stex_statements_sparagraph_end:
5585 }{
5586   \l_tmpa_tl
5587 }
5588 \end{stex_annotate_env}
5589 }
5590 }

```

\stexpatchparagraph

```

5591
5592 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5593   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5594     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5595       \titleemph{\l_stex_sparagraph_title_tl}:~
5596     }
5597   }{
5598     \titleemph{\l_stex_sparagraph_start_tl}~
5599   }
5600 }
5601 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5602
5603 \newcommand\stexpatchparagraph[3] [] {
5604   \str_set:Nx \l_tmpa_str{ #1 }
5605   \str_if_empty:NTF \l_tmpa_str {
5606     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5607     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5608   }{
5609     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5610     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5611   }
5612 }
5613
5614 \keys_define:nn { stex / inlinepara } {
5615   id      .str_set_x:N = \sparagraphid ,
5616   type    .str_set_x:N = \sparagraphtype ,
5617   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5618   from    .tl_set:N    = \sparagraphfrom ,
5619   to      .tl_set:N    = \sparagraphto ,
5620   name    .str_set:N   = \sparagraphname
5621 }
5622 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5623   \tl_clear:N \sparagraphfrom
5624   \tl_clear:N \sparagraphto
5625   \str_clear:N \sparagraphid
5626   \str_clear:N \sparagraphtype
5627   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5628   \str_clear:N \sparagraphname
5629   \keys_set:nn { stex / inlinepara }{ #1 }
5630 }
5631 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

5632 \begingroup
5633 \__stex_statements_inlinepara_args:n{ #1 }
5634 \clist_set:No \l_tmpa_clist \sparagraphtype
5635 \str_if_empty:NTF \sparagraphid {
5636   \str_if_empty:NTF \sparagraphname {
5637     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5638       \stex_ref_new_doc_target:n {}
5639     }
5640   } {
5641     \stex_ref_new_doc_target:n {}
5642   }
5643 } {
5644   \stex_ref_new_doc_target:n \sparagraphid
5645 }
5646 \stex_if_smsmode:TF{
5647   \str_if_empty:NF \sparagraphname {
5648     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5649   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5650 }
5651 }{
5652   \seq_clear:N \l_tmpa_seq
5653   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5654     \tl_if_empty:nF{ ##1 }{
5655       \stex_get_symbol:n { ##1 }
5656       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5657         \l_stex_get_symbol_uri_str
5658       }
5659     }
5660   }
5661   \exp_args:Nnx
5662   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5663     \str_if_empty:NF \sparagraphtype {
5664       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5665     }
5666     \str_if_empty:NF \sparagraphfrom {
5667       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5668     }
5669     \str_if_empty:NF \sparagraphto {
5670       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5671     }
5672     \str_if_empty:NF \sparagraphname {
5673       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5674     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5675     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5676   }
5677   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5678     \clist_map_inline:Nn \l_tmpa_seq {
5679       \stex_ref_new_sym_target:n {##1}
5680     }
5681   }
5682   #2
5683 }
5684 }
5685 \endgroup

```

```

5686 \stex_smsmode_do:
5687 }
5688
5689 (End definition for \stexpatchparagraph. This function is documented on page 42.)
5689 \endpackage

```

Chapter 33

The Implementation

33.1 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).⁸

```
5690 <*package>
5691 <@@=stex_sproof>
5692
5693 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5694
```

33.2 Proofs

We first define some keys for the proof environment.

```
5695 \keys_define:nn { stex / spf } {
5696   id          .str_set_x:N = \spfid,
5697   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5698   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5699   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5700   type        .str_set_x:N = \spftype,
5701   title       .tl_set:N    = \spftitle,
5702   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5703   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5704   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5705 }
5706 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5707   \str_clear:N \spfid
5708   \tl_clear:N \l__stex_sproof_spf_for_tl
5709   \tl_clear:N \l__stex_sproof_spf_from_tl
5710   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5711   \str_clear:N \spftype
5712   \tl_clear:N \spftitle
5713   \tl_clear:N \l__stex_sproof_spf_continues_tl
5714   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

⁸EdNOTE: need an implementation for L^AT_EXML

```

5715 \tl_clear:N \l__stex_sproof_spf_method_tl
5716 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5717 \keys_set:nn { stex / spf }{ #1 }
5718 }

```

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

```

5719 \str_set:Nn\c__stex_sproof_flow_str{inline}

```

(End definition for `\c__stex_sproof_flow_str`.)

For proofs, we will have to have deeply nested structures of enumerated list-like environments. However, L^AT_EX only allows `enumerate` environments up to nesting depth 4 and general list environments up to listing depth 6. This is not enough for us. Therefore we have decided to go along the route proposed by Leslie Lamport to use a single top-level list with dotted sequences of numbers to identify the position in the proof tree. Unfortunately, we could not use his `pf.sty` package directly, since it does not do automatic numbering, and we have to add keyword arguments all over the place, to accomodate semantic information.

`pst@with@label` This environment manages⁷ the path labeling of the proof steps in the description environment of the outermost `proof` environment. The argument is the label prefix up to now; which we cache in `\pst@label` (we need evaluate it first, since are in the right place now!). Then we increment the proof depth which is stored in `\count10` (lower counters are used by T_EX for page numbering) and initialize the next level counter `\count\count10` with 1. In the end call for this environment, we just decrease the proof depth counter by 1 again.

```

5720 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5721 \cs_new_protected:Npn \sproofnumber {
5722   \int_set:Nn \l_tmpa_int {1}
5723   \bool_while_do:nn {
5724     \int_compare_p:nNn {
5725       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5726     } > 0
5727   }{
5728     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5729     \int_incr:N \l_tmpa_int
5730   }
5731 }
5732 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5733   \int_set:Nn \l_tmpa_int {1}
5734   \bool_while_do:nn {
5735     \int_compare_p:nNn {
5736       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5737     } > 0
5738   }{
5739     \int_incr:N \l_tmpa_int
5740   }
5741   \int_compare:nNnF \l_tmpa_int = 1 {
5742     \int_decr:N \l_tmpa_int
5743   }
5744   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5745     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

⁷This gets the labeling right but only works 8 levels deep


```

5790     }
5791     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5792       \input{sproof-finnish.ldf}
5793     }
5794     \clist_if_in:NnT \l_tmpa_clist {french}{
5795       \input{sproof-french.ldf}
5796     }
5797     \clist_if_in:NnT \l_tmpa_clist {russian}{
5798       \input{sproof-russian.ldf}
5799     }
5800     \makeatother
5801   }{}
5802 }

```

spfsketch

```

5803 \newcommand\spfsketch[2] [] {
5804   \beginingroup
5805   \let \premise \stex_proof_premise:
5806   \__stex_sproof_spf_args:n{#1}
5807   \stex_if_smsmode:TF {
5808     \str_if_empty:NF \spfid {
5809       \stex_ref_new_doc_target:n \spfid
5810     }
5811   }{
5812     \seq_clear:N \l_tmpa_seq
5813     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5814       \tl_if_empty:nF{ ##1 }{
5815         \stex_get_symbol:n { ##1 }
5816         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5817           \l_stex_get_symbol_uri_str
5818         }
5819       }
5820     }
5821     \exp_args:Nnx
5822     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5823       \str_if_empty:NF \spftype {
5824         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5825       }
5826       \clist_set:No \l_tmpa_clist \spftype
5827       \tl_set:Nn \l_tmpa_tl {
5828         \titleemph{
5829           \tl_if_empty:NTF \spftitle {
5830             \spf@proofsketch@kw
5831           }{
5832             \spftitle
5833           }
5834         }::~
5835       }
5836       \clist_map_inline:Nn \l_tmpa_clist {
5837         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5838           \tl_clear:N \l_tmpa_tl
5839         }
5840       }
5841       \str_if_empty:NF \spfid {

```



```

5842         \stex_ref_new_doc_target:n \spfid
5843     }
5844     \l_tmpa_tl #2 \sproofend
5845 }
5846 }
5847 \endgroup
5848 \stex_smsmode_do:
5849 }
5850

```

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

EdN:9
EdN:10

spfeq This is very similar to \spfsketch, but uses a computation array⁹¹⁰

```

5851 \newenvironment{spfeq}[2][]{
5852   \__stex_sproof_spf_args:n{#1}
5853   \let \premise \stex_proof_premise:
5854   \stex_if_smsmode:TF {
5855     \str_if_empty:NF \spfid {
5856       \stex_ref_new_doc_target:n \spfid
5857     }
5858   }{
5859     \seq_clear:N \l_tmpa_seq
5860     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5861       \tl_if_empty:NF{ ##1 }{
5862         \stex_get_symbol:n { ##1 }
5863         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5864           \l_stex_get_symbol_uri_str
5865         }
5866       }
5867     }
5868     \exp_args:Nnnx
5869     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5870     \str_if_empty:NF \spftype {
5871       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5872     }
5873
5874     \clist_set:No \l_tmpa_clist \spftype
5875     \tl_clear:N \l_tmpa_tl
5876     \clist_map_inline:Nn \l_tmpa_clist {
5877       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5878         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5879       }
5880       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5881         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5882       }
5883     }
5884     \tl_if_empty:NTF \l_tmpa_tl {
5885       \__stex_sproof_spfeq_start:
5886     }{
5887       \l_tmpa_tl
5888     }{-#2}

```

⁹EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

¹⁰EDNOTE: document above

```

5889 \str_if_empty:NF \spfid {
5890 \stex_ref_new_doc_target:n \spfid
5891 }
5892 \begin{displaymath}\begin{array}{rc1l}
5893 }
5894 \stex_smsmode_do:
5895 }{
5896 \stex_if_smsmode:F {
5897 \end{array}\end{displaymath}
5898 \clist_set:No \l_tmpa_clist \spftype
5899 \tl_clear:N \l_tmpa_tl
5900 \clist_map_inline:Nn \l_tmpa_clist {
5901 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5902 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5903 }
5904 }
5905 \tl_if_empty:NTF \l_tmpa_tl {
5906 \__stex_sproof_spfeq_end:
5907 }{
5908 \l_tmpa_tl
5909 }
5910 \end{stex_annotate_env}
5911 }
5912 }
5913
5914 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5915 \titleemph{
5916 \tl_if_empty:NTF \spftitle {
5917 \spf@proof@kw
5918 }{
5919 \spftitle
5920 }
5921 }:
5922 }
5923 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5924
5925 \newcommand\stexpatchspfeq[3] [] {
5926 \str_set:Nx \l_tmpa_str{ #1 }
5927 \str_if_empty:NTF \l_tmpa_str {
5928 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5929 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5930 }{
5931 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5932 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5933 }
5934 }
5935

```

(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.

```

5936 \newenvironment{sproof}[2] []{

```

```

5937 \let \premise \stex_proof_premise:
5938 \intarray_gzero:N \l__stex_sproof_counter_intarray
5939 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5940 \__stex_sproof_spf_args:n{#1}
5941 \stex_if_smsmode:TF {
5942   \str_if_empty:NF \spfid {
5943     \stex_ref_new_doc_target:n \spfid
5944   }
5945 }{
5946   \seq_clear:N \l_tmpa_seq
5947   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5948     \tl_if_empty:NF{ ##1 }{
5949       \stex_get_symbol:n { ##1 }
5950       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5951         \l_stex_get_symbol_uri_str
5952       }
5953     }
5954   }
5955   \exp_args:Nnnx
5956   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5957   \str_if_empty:NF \spftype {
5958     \stex_annotate_invisible:nnn{type}{\spftype}{}
5959   }
5960
5961   \clist_set:No \l_tmpa_clist \spftype
5962   \tl_clear:N \l_tmpa_tl
5963   \clist_map_inline:Nn \l_tmpa_clist {
5964     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5965       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5966     }
5967     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5968       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5969     }
5970   }
5971   \tl_if_empty:NTF \l_tmpa_tl {
5972     \__stex_sproof_sproof_start:
5973   }{
5974     \l_tmpa_tl
5975   }{~#2}
5976   \str_if_empty:NF \spfid {
5977     \stex_ref_new_doc_target:n \spfid
5978   }
5979   \begin{description}
5980 }
5981 \stex_smsmode_do:
5982 }{
5983   \stex_if_smsmode:F{
5984     \end{description}
5985     \clist_set:No \l_tmpa_clist \spftype
5986     \tl_clear:N \l_tmpa_tl
5987     \clist_map_inline:Nn \l_tmpa_clist {
5988       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5989         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5990       }

```

```

5991     }
5992     \tl_if_empty:NTF \l_tmpa_tl {
5993       \__stex_sproof_sproof_end:
5994     }{
5995       \l_tmpa_tl
5996     }
5997     \end{stex_annotate_env}
5998   }
5999 }
6000
6001 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
6002   \par\noindent\titleemph{
6003     \tl_if_empty:NTF \spftype {
6004       \spf@proof@kw
6005     }{
6006       \spftype
6007     }
6008   }:
6009 }
6010 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
6011
6012 \newcommand\stexpatchproof[3] [] {
6013   \str_set:Nx \l_tmpa_str{ #1 }
6014   \str_if_empty:NTF \l_tmpa_str {
6015     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
6016     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
6017   }{
6018     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
6019     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
6020   }
6021 }

```

\spfidea

```

6022 \newcommand\spfidea[2] []{
6023   \__stex_sproof_spf_args:n{#1}
6024   \titleemph{
6025     \tl_if_empty:NTF \spftype {Proof~Idea}{
6026       \spftype
6027     }:
6028   }~#2
6029   \sproofend
6030 }

```

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

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

```

6031 \newenvironment{spfstep}[1] []{
6032   \__stex_sproof_spf_args:n{#1}
6033   \stex_if_smsmode:TF {

```

```

6034 \str_if_empty:NF \spfid {
6035   \stex_ref_new_doc_target:n \spfid
6036 }
6037 }{
6038   \@in@omtexttrue
6039   \seq_clear:N \l_tmpa_seq
6040   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6041     \tl_if_empty:nF{ ##1 }{
6042       \stex_get_symbol:n { ##1 }
6043       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6044         \l_stex_get_symbol_uri_str
6045       }
6046     }
6047   }
6048   \exp_args:Nnnx
6049   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
6050   \str_if_empty:NF \spftype {
6051     \stex_annotate_invisible:nnn{type}{\spftype}{}
6052   }
6053   \clist_set:No \l_tmpa_clist \spftype
6054   \tl_set:Nn \l_tmpa_tl {
6055     \item[\sproofnumber]
6056     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6057   }
6058   \clist_map_inline:Nn \l_tmpa_clist {
6059     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6060       \tl_clear:N \l_tmpa_tl
6061     }
6062   }
6063   \l_tmpa_tl
6064   \tl_if_empty:NF \spftitle {
6065     {(\titleemph{\spftitle})\enspace}
6066   }
6067   \str_if_empty:NF \spfid {
6068     \stex_ref_new_doc_target:n \spfid
6069   }
6070 }
6071 \stex_smsmode_do:
6072 \ignorespacesandpars
6073 }{
6074   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6075     \__stex_sproof_inc_counter:
6076   }
6077   \stex_if_smsmode:F {
6078     \end{stex_annotate_env}
6079   }
6080 }

```

sproofcomment

```

6081 \newenvironment{sproofcomment}[1][]{
6082   \__stex_sproof_spf_args:n{#1}
6083   \clist_set:No \l_tmpa_clist \spftype
6084   \tl_set:Nn \l_tmpa_tl {
6085     \item[\sproofnumber]

```

```

6086 \bool_set_true:N \l__stex_sproof_inc_counter_bool
6087 }
6088 \clist_map_inline:Nn \l_tmpa_clist {
6089 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6090 \tl_clear:N \l_tmpa_tl
6091 }
6092 }
6093 \l_tmpa_tl
6094 }{
6095 \bool_if:NT \l__stex_sproof_inc_counter_bool {
6096 \__stex_sproof_inc_counter:
6097 }
6098 }

```

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.

```

6099 \newenvironment{subproof}[2][]{
6100 \__stex_sproof_spf_args:n{#1}
6101 \stex_if_smsmode:TF{
6102 \str_if_empty:NF \spfid {
6103 \stex_ref_new_doc_target:n \spfid
6104 }
6105 }{
6106 \seq_clear:N \l_tmpa_seq
6107 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6108 \tl_if_empty:nF{ ##1 }{
6109 \stex_get_symbol:n { ##1 }
6110 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6111 \l_stex_get_symbol_uri_str
6112 }
6113 }
6114 }
6115 \exp_args:Nnnx
6116 \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
6117 \str_if_empty:NF \spftype {
6118 \stex_annotate_invisible:nnn{type}{\spftype}{}}
6119 }
6120
6121 \clist_set:No \l_tmpa_clist \spftype
6122 \tl_set:Nn \l_tmpa_tl {
6123 \item[\sproofnumber]
6124 \bool_set_true:N \l__stex_sproof_inc_counter_bool
6125 }
6126 \clist_map_inline:Nn \l_tmpa_clist {
6127 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6128 \tl_clear:N \l_tmpa_tl
6129 }
6130 }
6131 \l_tmpa_tl
6132 \tl_if_empty:NF \spftitle {
6133 {(\titleemph{\spftitle})\enspace}
6134 }

```

```

6135     {~#2}
6136     \str_if_empty:NF \spfid {
6137       \stex_ref_new_doc_target:n \spfid
6138     }
6139   }
6140   \__stex_sproof_add_counter:
6141   \stex_smsmode_do:
6142 }{
6143   \__stex_sproof_remove_counter:
6144   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6145     \__stex_sproof_inc_counter:
6146   }
6147   \stex_if_smsmode:F{
6148     \end{stex_annotate_env}
6149   }
6150 }

```

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

```

6151 \newenvironment{spfcases}[2][]{
6152   \tl_if_empty:nTF{#1}{
6153     \begin{subproof}[method=by-cases]{#2}
6154   }{
6155     \begin{subproof}[#1,method=by-cases]{#2}
6156   }
6157 }{
6158   \end{subproof}
6159 }

```

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

```

6160 \newenvironment{spfcase}[2][]{
6161   \__stex_sproof_spf_args:n{#1}
6162   \stex_if_smsmode:TF {
6163     \str_if_empty:NF \spfid {
6164       \stex_ref_new_doc_target:n \spfid
6165     }
6166   }{
6167     \seq_clear:N \l_tmpa_seq
6168     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6169       \tl_if_empty:nF{ ##1 }{
6170         \stex_get_symbol:n { ##1 }
6171         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6172           \l_stex_get_symbol_uri_str
6173         }
6174       }
6175     }
6176     \exp_args:Nnnx
6177     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6178     \str_if_empty:NF \spftype {
6179       \stex_annotate_invisible:nnn{type}{\spftype}{}}
6180   }
6181   \clist_set:Nn \l_tmpa_clist \spftype
6182   \tl_set:Nn \l_tmpa_tl {
6183     \item[\sproofnumber]

```

```

6184     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6185   }
6186   \clist_map_inline:Nn \l_tmpa_clist {
6187     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6188       \tl_clear:N \l_tmpa_tl
6189     }
6190   }
6191   \l_tmpa_tl
6192   \tl_if_empty:nF{#2}{
6193     \titleemph{#2}:~
6194   }
6195 }
6196 \__stex_sproof_add_counter:
6197 \stex_smsmode_do:
6198 ){
6199   \__stex_sproof_remove_counter:
6200   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6201     \__stex_sproof_inc_counter:
6202   }
6203   \stex_if_smsmode:F{
6204     \clist_set:No \l_tmpa_clist \spftype
6205     \tl_set:Nn \l_tmpa_tl{\sproofend}
6206     \clist_map_inline:Nn \l_tmpa_clist {
6207       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6208         \tl_clear:N \l_tmpa_tl
6209       }
6210     }
6211     \l_tmpa_tl
6212     \end{stex_annotate_env}
6213   }
6214 }

```

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

```

6215 \newcommand\spfcasesketch[3][]{
6216   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6217 }

```

33.3 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.

```

6218 \keys_define:nn { stex / just }{
6219   id      .str_set:x:N = \l__stex_sproof_just_id_str,
6220   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
6221   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
6222   args     .tl_set:N    = \l__stex_sproof_just_args_tl
6223 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.¹¹

¹¹EDNOTE: need to do something about the premise in draft mode.

justification

```
6224 \newenvironment{justification}[1] [] {}{}
```

\premise

```
6225 \newcommand\stex_proof_promise:[2] [] {#2}
```

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

\justarg the **\justarg** macro is purely semantic, so we ignore the keyval arguments for now and only display the content.

```
6226 \newcommand\justarg[2] [] {#2}
```

```
6227 \end{package}
```

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

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 34

STEX -Others Implementation

```
6228 <*package>
6229
6230 %%%%%%%%%% others.dtx %%%%%%%%%%
6231
6232 <@@=stex_others>
    Warnings and error messages
6233 % None

\MSC Math subject classifier

6234 \NewDocumentCommand \MSC {m} {
6235 % TODO
6236 }

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

6237 \@ifpackageloaded{tikzinput}{
6238 \RequirePackage{stex-tikzinput}
6239 }{}

6240 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6241 <*package>
6242 <@@=stex_modules>
6243
6244 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6245
6246 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
6247 \begingroup
6248 \stex_module_setup:nn{
6249   ns=\c_stex_metatheory_ns_str,
6250   meta=NONE
6251 }{Metatheory}
6252 \stex_reactivate_macro:N \symdecl
6253 \stex_reactivate_macro:N \notation
6254 \stex_reactivate_macro:N \symdef
6255 \ExplSyntaxOff
6256 \csname stex_suppress_html:n\endcsname{
6257   % is-a (a:A, a \in A, a is an A, etc.)
6258   \symdecl{isa}[args=ai]
6259   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6260   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6261   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6262
6263   % bind (\forall, \Pi, \lambda etc.)
6264   \symdecl{bind}[args=Bi]
6265   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6266   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6267   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
6268
6269   % implicit bind
6270   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6271
6272   % dummy variable
6273   \symdecl{dummyvar}
6274   \notation{dummyvar}[underscore]{\comp\_}
6275   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6276 \notation{dummyvar}[dash]{\comp{\rm --}}
6277
6278 %fromto (function space, Hom-set, implication etc.)
6279 \symdecl{fromto}[args=ai]
6280 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6281 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6282
6283 % mapto (lambda etc.)
6284 \symdecl{mapto}[args=Bi]
6285 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6286 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6287 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6288
6289 % function/operator application
6290 \symdecl{apply}[args=ia]
6291 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6292 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6293
6294 % collection of propositions/booleans/truth values
6295 \symdecl{prop}[name=proposition]
6296 \notation{prop}[prop]{\comp{\rm prop}}
6297 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6298
6299 \symdecl{judgmentholds}[args=1]
6300 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6301
6302 % sequences
6303 \symdecl{seqtype}[args=1]
6304 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6305
6306 \symdecl{seqexpr}[args=a]
6307 \notation{seqexpr}[angle,prec=nobrackets]{\comp\angle #1\comp\rangle}{##1\comp,##2}
6308
6309 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6310 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6311
6312 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6313 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6314 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
6315
6316 % letin (''let'', local definitions, variable substitution)
6317 \symdecl{letin}[args=bii]
6318 \notation{letin}[let]{\comp{\rm let}}\; ;#1\comp{=}\; #2\; ;\comp{\rm in}}\; ;#3}
6319 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6320 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6321
6322 % structures
6323 \symdecl*{module-type}[args=1]
6324 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6325 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6326 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6327
6328 % objects
6329 \symdecl{object}

```

```

6330 \notation{object}{\comp{\mathtt{OBJECT}}}
6331
6332 }
6333 \ExplSyntaxOn
6334 \stex_add_to_current_module:n{
6335   \let\nappa\apply
6336   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6337   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6338   \def\livar{\csname sequence-index\endcsname[li]}
6339   \def\uivar{\csname sequence-index\endcsname[ui]}
6340   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6341   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6342   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6343 }
6344 \__stex_modules_end_module:
6345 \endgroup
6346 \</package>

```

Chapter 36

Tikzinput Implementation

```
6347 <*package>
6348
6349 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6350
6351 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6352 \RequirePackage{l3keys2e}
6353
6354 \keys_define:nn { tikzinput } {
6355   image .bool_set:N = \c_tikzinput_image_bool,
6356   image .default:n = false ,
6357   unknown .code:n = {}
6358 }
6359
6360 \ProcessKeysOptions { tikzinput }
6361
6362 \bool_if:NTF \c_tikzinput_image_bool {
6363   \RequirePackage{graphicx}
6364
6365   \providecommand\usetikzlibrary[]{}
6366   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6367 }{
6368   \RequirePackage{tikz}
6369   \RequirePackage{standalone}
6370
6371   \newcommand \tikzinput [2] [] {
6372     \setkeys{Gin}{#1}
6373     \ifx \Gin@ewidth \Gin@exclamation
6374       \ifx \Gin@eheight \Gin@exclamation
6375         \input { #2 }
6376       \else
6377         \resizebox{!}{ \Gin@eheight }{
6378           \input { #2 }
6379         }
6380       \fi
6381     \else
6382       \ifx \Gin@eheight \Gin@exclamation
6383         \resizebox{ \Gin@ewidth }{!}{
6384           \input { #2 }
```

```

6385     }
6386     \else
6387         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6388             \input { #2 }
6389         }
6390     \fi
6391 \fi
6392 }
6393 }
6394
6395 \newcommand \ctikzinput [2] [] {
6396     \begin{center}
6397         \tikzinput [1] {#2}
6398     \end{center}
6399 }
6400
6401 \@ifpackageloaded{stex}{
6402     \RequirePackage{stex-tikzinput}
6403 }{}
6404
6405 </package>
6406 <*stex>
6407 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6408 \RequirePackage{stex}
6409 \RequirePackage{tikzinput}
6410
6411 \newcommand\mhtikzinput [2] [] {%
6412     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6413     \stex_in_repository:nn\Gin@mhrepos{
6414         \tikzinput [1]{\mhpath{##1}{#2}}
6415     }
6416 }
6417 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
6418 </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

37.1 The document-structure Class

The functionality is spread over the `document-structure` class and package. The class provides the `document` environment and the `document-structure` element corresponds to it, whereas the package provides the concrete functionality.

```
6419 \*cls)
6420 \@@=document_structure)
6421 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6422 \RequirePackage{13keys2e}
```

37.2 Class Options

To initialize the `document-structure` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
6423 \keys_define:nn{ document-structure / pkg }{
6424   class      .str_set_x:N = \c_document_structure_class_str,
6425   minimal    .bool_set:N = \c_document_structure_minimal_bool,
6426   report     .code:n      = {
6427     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6428     \str_set:Nn \c_document_structure_class_str {report}
6429   },
6430   book       .code:n      = {
6431     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6432     \str_set:Nn \c_document_structure_class_str {book}
6433   },
6434   bookpart   .code:n      = {
6435     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6436     \str_set:Nn \c_document_structure_class_str {book}
6437     \str_set:Nn \c_document_structure_topsect_str {chapter}
6438   },
```



```

6439 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6440 unknown     .code:n      = {
6441   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6442 }
6443 }
6444 \ProcessKeysOptions{ document-structure / pkg }
6445 \str_if_empty:NT \c_document_structure_class_str {
6446   \str_set:Nn \c_document_structure_class_str {article}
6447 }
6448 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6449   {\c_document_structure_class_str}
6450

```

37.3 Beefing up the document environment

Now, – unless the option `minimal` is defined – we include the `stex` package

```

6451 \RequirePackage{document-structure}
6452 \bool_if:NF \c_document_structure_minimal_bool {

```

And define the environments we need. The top-level one is the `document` environment, which we redefined so that we can provide keyval arguments.

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.¹²

```

6453 \keys_define:nn { document-structure / document }{
6454   id .str_set_x:N = \c_document_structure_document_id_str
6455 }
6456 \let\__document_structure_orig_document=\document
6457 \renewcommand{\document}[1][]{
6458   \keys_set:nn{ document-structure / document }{ #1 }
6459   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6460   \__document_structure_orig_document
6461 }

```

Finally, we end the test for the `minimal` option.

```

6462 }
6463 \</cls>

```

37.4 Implementation: document-structure Package

```

6464 \<*package>
6465 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6466 \RequirePackage{l3keys2e}

```

37.5 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).

¹²EDNOTE: faking documentkeys for now. @HANG, please implement

```

6467
6468 \keys_define:nn{ document-structure / pkg }{
6469   class      .str_set_x:N = \c_document_structure_class_str,
6470   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6471   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6472 }
6473 \ProcessKeysOptions{ document-structure / pkg }
6474 \str_if_empty:NT \c_document_structure_class_str {
6475   \str_set:Nn \c_document_structure_class_str {article}
6476 }
6477 \str_if_empty:NT \c_document_structure_topsect_str {
6478   \str_set:Nn \c_document_structure_topsect_str {section}
6479 }

```

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

```

6480 \RequirePackage{xspace}
6481 \RequirePackage{comment}
6482 \AddToHook{begindocument}{
6483   \ltx@ifpackageloaded{babel}{
6484     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6485     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6486       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6487     }
6488   }{}
6489 }

```

`\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.

```

6490 \int_new:N \l_document_structure_section_level_int
6491 \str_case:VnF \c_document_structure_topsect_str {
6492   {part}}{
6493     \int_set:Nn \l_document_structure_section_level_int {0}
6494   }
6495   {chapter}{
6496     \int_set:Nn \l_document_structure_section_level_int {1}
6497   }
6498 }{
6499   \str_case:VnF \c_document_structure_class_str {
6500     {book}}{
6501       \int_set:Nn \l_document_structure_section_level_int {0}
6502     }
6503     {report}}{
6504       \int_set:Nn \l_document_structure_section_level_int {0}
6505     }
6506   }{
6507     \int_set:Nn \l_document_structure_section_level_int {2}
6508   }
6509 }

```

37.6 Document Structure

The structure of the document is given by the `omgroup` environment just like in OMDoc. 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.¹³

EdN:13

```
6510 \def\current@section@level{document}%
6511 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6512 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6513 \cs_new_protected:Npn \skipomgroup {
6514   \ifcase\l_document_structure_section_level_int
6515   \or\stepcounter{part}
6516   \or\stepcounter{chapter}
6517   \or\stepcounter{section}
6518   \or\stepcounter{subsection}
6519   \or\stepcounter{subsubsection}
6520   \or\stepcounter{paragraph}
6521   \or\stepcounter{subparagraph}
6522   \fi
6523 }
```

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

`blindfragment`

```
6524 \newcommand\at@begin@blindomgroup[1]{%
6525   \newenvironment{blindfragment}
6526   {
6527     \int_incr:N\l_document_structure_section_level_int
6528     \at@begin@blindomgroup\l_document_structure_section_level_int
6529   }{}}
```

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

```
6530 \newcommand\omgroup@nonum[2]{
6531   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6532   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6533 }
```

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

`\omgroup@num` convenience macro: `\omgroup@num{<level>}{<title>}` makes numbered sectioning with title `<title>` at level `<level>`. We have to check the `short` key was given in the `omgroup` 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.

```
6534 \newcommand\omgroup@num[2]{
```

¹³EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

6535 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6536   \@nameuse{#1}{#2}
6537 }{
6538   \cs_if_exist:NTF\rdfmata@sectioning{
6539     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6540   }{
6541     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6542   }
6543 }
6544 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6545 }

```

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

sfragment

```

6546 \keys_define:nn { document-structure / omgroup }{
6547   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6548   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6549   creators     .clist_set:N = \l__document_structure_omgroup_creators_clist,
6550   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6551   srccite      .tl_set:N   = \l__document_structure_omgroup_srccite_tl,
6552   type         .tl_set:N   = \l__document_structure_omgroup_type_tl,
6553   short        .tl_set:N   = \l__document_structure_omgroup_short_tl,
6554   display      .tl_set:N   = \l__document_structure_omgroup_display_tl,
6555   intro        .tl_set:N   = \l__document_structure_omgroup_intro_tl,
6556   loadmodules  .bool_set:N = \l__document_structure_omgroup_loadmodules_bool
6557 }
6558 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6559   \str_clear:N \l__document_structure_omgroup_id_str
6560   \str_clear:N \l__document_structure_omgroup_date_str
6561   \clist_clear:N \l__document_structure_omgroup_creators_clist
6562   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6563   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6564   \tl_clear:N \l__document_structure_omgroup_type_tl
6565   \tl_clear:N \l__document_structure_omgroup_short_tl
6566   \tl_clear:N \l__document_structure_omgroup_display_tl
6567   \tl_clear:N \l__document_structure_omgroup_intro_tl
6568   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6569   \keys_set:nn { document-structure / omgroup } { #1 }
6570 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The \at@begin@omgroup macro allows customization. It is run at the beginning of the omgroup, i.e. after the section heading.

```

6571 \newif\if@mainmatter\@mainmattertrue
6572 \newcommand\at@begin@omgroup[3] [] {}

```

Then we define a helper macro that takes care of the sectioning magic. It comes with its own key/value interface for customization.

```

6573 \keys_define:nn { document-structure / sectioning }{
6574   name .str_set_x:N = \l__document_structure_sect_name_str ,
6575   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6576   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6577   clear .default:n = {true} ,
6578   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6579   num      .default:n      = {true}
6580 }
6581 \cs_new_protected:Nn \__document_structure_sect_args:n {
6582   \str_clear:N \l__document_structure_sect_name_str
6583   \str_clear:N \l__document_structure_sect_ref_str
6584   \bool_set_false:N \l__document_structure_sect_clear_bool
6585   \bool_set_false:N \l__document_structure_sect_num_bool
6586   \keys_set:nn { document-structure / sectioning } { #1 }
6587 }
6588 \newcommand\omdoc@sectioning[3][]{
6589   \__document_structure_sect_args:n {#1}
6590   \let\omdoc@sect@name\l__document_structure_sect_name_str
6591   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6592   \if@mainmatter% numbering not overridden by frontmatter, etc.
6593     \bool_if:NTF \l__document_structure_sect_num_bool {
6594       \omgroup@num{#2}{#3}
6595     }{
6596       \omgroup@nonum{#2}{#3}
6597     }
6598     \def\current@section@level{\omdoc@sect@name}
6599   \else
6600     \omgroup@nonum{#2}{#3}
6601   \fi
6602 }% 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.

```

6603 \newcommand\omgroup@redefine@addtocontents[1]{%
6604 %\edef\__document_structureimport{#1}%
6605 %\@for\@I:=\__document_structureimport\do{%
6606 %\edef\@path{\csname module@\@I @path\endcsname}%
6607 %\@ifundefined{tf@toc}\relax%
6608 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6609 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6610 %\def\addcontentsline##1##2##3{%
6611 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6612 %\else% hyperref.sty not loaded
6613 %\def\addcontentsline##1##2##3{%
6614 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6615 %\fi
6616 }% hypreref.sty loaded?

```

now the `omgroup` 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 `omgroups` in the `\omgroup@level` counter.

```

6617 \newenvironment{sfragment}[2][]{% keys, title
6618 {
6619   \__document_structure_omgroup_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.

```

6620 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6621   \omgroup@redefine@addtocontents{
6622     \@ifundefined{module@id}\used@modules%

```

```

6623     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6624   }
6625 }

```

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

```

6626 \int_incr:N\l_document_structure_section_level_int
6627 \ifcase\l_document_structure_section_level_int
6628   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6629   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6630   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6631   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6632   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6633   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6634   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6635 \fi
6636 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6637 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6638   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6639 }
6640 }% for customization
6641 {}

```

and finally, we localize the sections

```

6642 \newcommand\omdoc@part@kw{Part}
6643 \newcommand\omdoc@chapter@kw{Chapter}
6644 \newcommand\omdoc@section@kw{Section}
6645 \newcommand\omdoc@subsection@kw{Subsection}
6646 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6647 \newcommand\omdoc@paragraph@kw{paragraph}
6648 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

37.7 Front and Backmatter

Index markup is provided by the `omtext` package [Koh20c], so in the `document-structure` package we only need to supply the corresponding `\printindex` command, if it is not already defined

`\printindex`

```

6649 \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).

```

6650 \cs_if_exist:NTF\frontmatter{
6651   \let\__document_structure_orig_frontmatter\frontmatter
6652   \let\frontmatter\relax
6653 }{
6654   \tl_set:Nn\__document_structure_orig_frontmatter{
6655     \clearpage
6656     \@mainmatterfalse
6657     \pagenumbering{roman}

```

```

6658 }
6659 }
6660 \cs_if_exist:NTF\backmatter{
6661   \let\__document_structure_orig_backmatter\backmatter
6662   \let\backmatter\relax
6663 }{
6664   \tl_set:Nn\__document_structure_orig_backmatter{
6665     \clearpage
6666     \@mainmatterfalse
6667     \pagenumbering{roman}
6668   }
6669 }

```

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.

```

6670 \newenvironment{frontmatter}{
6671   \__document_structure_orig_frontmatter
6672 }{
6673   \cs_if_exist:NTF\mainmatter{
6674     \mainmatter
6675   }{
6676     \clearpage
6677     \@mainmattertrue
6678     \pagenumbering{arabic}
6679   }
6680 }

```

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

```

6681 \newenvironment{backmatter}{
6682   \__document_structure_orig_backmatter
6683 }{
6684   \cs_if_exist:NTF\mainmatter{
6685     \mainmatter
6686   }{
6687     \clearpage
6688     \@mainmattertrue
6689     \pagenumbering{arabic}
6690   }
6691 }

```

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

```

6692 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6693 \def \c__document_structure_document_str{document}
6694 \newcommand\afterprematurestop{}
6695 \def\prematurestop@endomgroup{
6696   \unless\ifx\@currenvir\c__document_structure_document_str
6697     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
6698     \expandafter\prematurestop@endomgroup

```

```

6699 \fi
6700 }
6701 \providecommand\prematurestop{
6702   \message{Stopping~sTeX~processing~prematurely}
6703   \prematurestop@endgroup
6704   \afterprematurestop
6705   \end{document}
6706 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

6707 \RequirePackage{etoolbox}
6708 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6709 \newrobustcmd\useSGvar[1]{%
6710   \@ifundefined{sTeX@Gvar@#1}
6711   {\PackageError{document-structure}
6712     {The sTeX Global variable #1 is undefined}
6713     {set it with \protect\setSGvar}}
6714   \@nameuse{sTeX@Gvar@#1}}

```

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

\ifSGvar execute something conditionally based on the state of the global variable.

```

6715 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6716   \@ifundefined{sTeX@Gvar@#1}
6717   {\PackageError{document-structure}
6718     {The sTeX Global variable #1 is undefined}
6719     {set it with \protect\setSGvar}}
6720   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

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.

```
6721 \*cls)
6722 \@@=notesslides}
6723 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6724 \RequirePackage{13keys2e}
6725
6726 \keys_define:nn{notesslides / cls}{
6727   class .code:n = {
6728     \PassOptionsToClass{\CurrentOption}{document-structure}
6729     \str_if_eq:nnT{#1}{book}{
6730       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6731     }
6732     \str_if_eq:nnT{#1}{report}{
6733       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6734     }
6735   },
6736   notes .bool_set:N = \c__notesslides_notes_bool ,
6737   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6738   unknown .code:n = {
6739     \PassOptionsToClass{\CurrentOption}{document-structure}
6740     \PassOptionsToClass{\CurrentOption}{beamer}
6741     \PassOptionsToPackage{\CurrentOption}{notesslides}
6742   }
6743 }
6744 \ProcessKeysOptions{ notesslides / cls }
6745 \bool_if:NTF \c__notesslides_notes_bool {
6746   \PassOptionsToPackage{notes=true}{notesslides}
6747 }{
6748   \PassOptionsToPackage{notes=false}{notesslides}
6749 }
6750 \</cls>
```

now we do the same for the notesslides package.

```

6751 <*package>
6752 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6753 \RequirePackage{13keys2e}
6754
6755 \keys_define:nn{notesslides / pkg}{
6756   topsect          .str_set_x:N = \c__notesslides_topsect_str,
6757   defaulttopsect   .str_set_x:N = \c__notesslides_defaulttopsec_str,
6758   notes            .bool_set:N = \c__notesslides_notes_bool ,
6759   slides           .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6760   sectocframes     .bool_set:N = \c__notesslides_sectocframes_bool ,
6761   frameimages      .bool_set:N = \c__notesslides_frameimages_bool ,
6762   fiboxed          .bool_set:N = \c__notesslides_fiboxed_bool ,
6763   noproblems       .bool_set:N = \c__notesslides_noproblems_bool,
6764   unknown          .code:n      = {
6765     \PassOptionsToClass{\CurrentOption}{stex}
6766     \PassOptionsToClass{\CurrentOption}{tikzinput}
6767   }
6768 }
6769 \ProcessKeysOptions{ notesslides / pkg }
6770 \newif\ifnotes
6771 \bool_if:NTF \c__notesslides_notes_bool {
6772   \notesttrue
6773 }{
6774   \notesfalse
6775 }
6776

```

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

```

6777 \str_if_empty:NTF \c__notesslides_topsect_str {
6778   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6779 }{
6780   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6781 }
6782 </package>

```

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

```

6783 <*cls>
6784 \bool_if:NTF \c__notesslides_notes_bool {
6785   \LoadClass{document-structure}
6786 }{
6787   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6788   \newcounter{Item}
6789   \newcounter{paragraph}
6790   \newcounter{subparagraph}
6791   \newcounter{Hfootnote}
6792   \RequirePackage{document-structure}
6793 }

```

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

```

6794 \RequirePackage{notesslides}
6795 </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 `STEX` packages. The first batch of packages we want are loaded on `notesslides.sty`. These are the general ones, we will load the `STEX`-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.

```

6796 \*package>
6797 \bool_if:NT \c__notesslides_notes_bool {
6798   \RequirePackage{a4wide}
6799   \RequirePackage{marginnote}
6800   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6801   \RequirePackage{mdframed}
6802   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6803   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6804 }
6805 \RequirePackage{stex-tikzinput}
6806 \RequirePackage{etoolbox}
6807 \RequirePackage{amssymb}
6808 \RequirePackage{amsmath}
6809 \RequirePackage{comment}
6810 \RequirePackage{textcomp}
6811 \RequirePackage{url}
6812 \RequirePackage{graphicx}
6813 \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`.¹⁴

```

6814 \bool_if:NT \c__notesslides_notes_bool {
6815   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6816 }
6817
6818
6819 \NewDocumentCommand \libusetheme {0{} m} {
6820   \bool_if:NTF \c__notesslides_notes_bool {
6821     \libusepackage[#1]{beamernotestheme#2}
6822   }{
6823     \libusepackage[#1]{beamertheme#2}
6824   }
6825 }

```

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

```

6826 \newcounter{slide}
6827 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6828 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

¹⁴EdNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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.

```

6829 \bool_if:NTF \c__notesslides_notes_bool {
6830   \renewenvironment{note}{\ignorespaces}{}
6831 }{
6832   \excludecomment{note}
6833 }

```

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.

```

6834 \bool_if:NT \c__notesslides_notes_bool {
6835   \newlength{\slideframewidth}
6836   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6837 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6838   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6839     \bool_set_true:N #1
6840   }{
6841     \bool_set_false:N #1
6842   }
6843 }
6844 \keys_define:nn{notesslides / frame}{
6845   label .str_set_x:N = \l__notesslides_frame_label_str,
6846   allowframebreaks .code:n = {
6847     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6848   },
6849   allowdisplaybreaks .code:n = {
6850     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6851   },
6852   fragile .code:n = {
6853     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6854   },
6855   shrink .code:n = {
6856     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6857   },
6858   squeeze .code:n = {
6859     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6860   },
6861   t .code:n = {
6862     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6863   },
6864 }
6865 \cs_new_protected:Nn \__notesslides_frame_args:n {
6866   \str_clear:N \l__notesslides_frame_label_str
6867   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6868   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6869   \bool_set_true:N \l__notesslides_frame_fragile_bool
6870   \bool_set_true:N \l__notesslides_frame_shrink_bool
6871   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6872   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6873 \keys_set:nn { notesslides / frame }{ #1 }
6874 }

```

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

```

6875 \renewenvironment{frame}[1][]{
6876 \_notesslides_frame_args:n{#1}
6877 \sffamily
6878 \stepcounter{slide}
6879 \def\@currentlabel{\theslide}
6880 \str_if_empty:NF \l__notesslides_frame_label_str {
6881 \label{\l__notesslides_frame_label_str}
6882 }

```

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

```

6883 \def\itemize@level{outer}
6884 \def\itemize@outer{outer}
6885 \def\itemize@inner{inner}
6886 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6887 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6888 \renewenvironment{itemize}{
6889 \ifx\itemize@level\itemize@outer
6890 \def\itemize@label{$\rhd$}
6891 \fi
6892 \ifx\itemize@level\itemize@inner
6893 \def\itemize@label{$\scriptstyle\rhd$}
6894 \fi
6895 \begin{list}
6896 {\itemize@label}
6897 {\setlength{\labelsep}{.3em}
6898 \setlength{\labelwidth}{.5em}
6899 \setlength{\leftmargin}{1.5em}
6900 }
6901 \edef\itemize@level{\itemize@inner}
6902 }{
6903 \end{list}
6904 }

```

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

```

6905 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6906 }{
6907 \medskip\miko@slidelabel\end{mdframed}
6908 }

```

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

`\frametitle`

```

6909 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6910 }

```

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

EdN:15

`\pause`

```

15
6911 \bool_if:NT \c__notesslides_notes_bool {
6912 \newcommand\pause{}
6913 }

```

¹⁵EdNOTE: MK: fake it in notes mode for now

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

nparagraph

```
6914 \bool_if:NTF \c__notesslides_notes_bool {
6915   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6916 }{
6917   \excludecomment{nparagraph}
6918 }
```

nfragment

```
6919 \bool_if:NTF \c__notesslides_notes_bool {
6920   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6921 }{
6922   \excludecomment{nfragment}
6923 }
```

ndefinition

```
6924 \bool_if:NTF \c__notesslides_notes_bool {
6925   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6926 }{
6927   \excludecomment{ndefinition}
6928 }
```

nassertion

```
6929 \bool_if:NTF \c__notesslides_notes_bool {
6930   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6931 }{
6932   \excludecomment{nassertion}
6933 }
```

nsproof

```
6934 \bool_if:NTF \c__notesslides_notes_bool {
6935   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6936 }{
6937   \excludecomment{nsproof}
6938 }
```

nexample

```
6939 \bool_if:NTF \c__notesslides_notes_bool {
6940   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6941 }{
6942   \excludecomment{nexample}
6943 }
```

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

```
6944 \def\inputref@preskip{\smallskip}
6945 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6946 \let\orig@inputref\inputref
6947 \def\inputref{\@ifstar\ninputref\orig@inputref}
6948 \newcommand\ninputref[2][]{
6949   \bool_if:NT \c__notesslides_notes_bool {
6950     \orig@inputref[#1]{#2}
6951   }
6952 }
```

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

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.

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

```
6953 \newlength{\slidelogoheight}
6954
6955 \bool_if:NTF \c__notesslides_notes_bool {
6956   \setlength{\slidelogoheight}{.4cm}
6957 }{
6958   \setlength{\slidelogoheight}{1cm}
6959 }
6960 \newsavebox{\slidelogo}
6961 \sbox{\slidelogo}{\text{\TeX}}
6962 \newrobustcmd{\setslidelogo}[1]{
6963   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6964 }
```

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

`\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.

```
6965 \def\source{Michael Kohlhase}% customize locally
6966 \newrobustcmd{\setsource}[1]{\def\source{#1}}
```

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

`\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.

```
6967 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6968 \newsavebox{\cclogo}
6969 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6970 \newif\ifcchref\cchreffalse
6971 \AtBeginDocument{
6972   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6973 }
6974 \def\licensing{
6975   \ifcchref
```

```

6976     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6977 \else
6978     {\usebox{\cclogo}}
6979 \fi
6980 }
6981 \newrobustcmd{\setlicensing}[2][]{
6982   \def\@url{#1}
6983   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6984   \ifx\@url\@empty
6985     \def\licensing{\usebox{\cclogo}}
6986   \else
6987     \def\licensing{
6988       \ifcchref
6989         \href{#1}{\usebox{\cclogo}}
6990       \else
6991         {\usebox{\cclogo}}
6992       \fi
6993     }
6994   \fi
6995 }

```

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

EdN:16

\slidelabel Now, we set up the slide label for the article mode.¹⁶

```

6996 \newrobustcmd\miko@slidelabel{
6997   \vbox to \slidelogoheight{
6998     \vss\hbox to \slidewidth
6999     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
7000   }
7001 }

```

(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.

```

7002 \def\Gin@mhrepos{}
7003 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
7004 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
7005 \newrobustcmd\frameimage[2][]{
7006   \stepcounter{slide}
7007   \bool_if:NT \c__notesslides_frameimages_bool {
7008     \def\Gin@ewidth{}\setkeys{Gin}{#1}
7009     \bool_if:NF \c__notesslides_notes_bool { \vfill }
7010     \begin{center}
7011       \bool_if:NTF \c__notesslides_fboxed_bool {
7012         \fbox{
7013           \ifx\Gin@ewidth\@empty
7014             \ifx\Gin@mhrepos\@empty
7015               \mhgraphics[width=\slidewidth,#1]{#2}
7016             \else

```

¹⁶EdNOTE: see that we can use the themes for the slides some day. This is all fake.


```

7017         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7018     \fi
7019 \else% Gin@ewidth empty
7020     \ifx\Gin@mhrepos\@empty
7021         \mhgraphics[#1]{#2}
7022     \else
7023         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7024     \fi
7025 \fi% Gin@ewidth empty
7026 }
7027 }{
7028     \ifx\Gin@ewidth\@empty
7029     \ifx\Gin@mhrepos\@empty
7030         \mhgraphics[width=\slidewidth,#1]{#2}
7031     \else
7032         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
7033     \fi
7034     \ifx\Gin@mhrepos\@empty
7035         \mhgraphics[#1]{#2}
7036     \else
7037         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
7038     \fi
7039 \fi% Gin@ewidth empty
7040 }
7041 \end{center}
7042 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
7043 \bool_if:NF \c__notesslides_notes_bool { \vfill }
7044 }
7045 } % ifmks@sty@frameimages

```

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

38.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

7046 \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

```

7047 \AddToHook{begindocument}{
7048     \definecolor{green}{rgb}{0,.5,0}
7049     \definecolor{purple}{cmyk}{.3,1,0,.17}
7050 }

```

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`.

```

7051 % \def\STpresent#1{\textcolor{blue}{#1}}
7052 \def\defemph#1{\textcolor{magenta}{#1}}
7053 \def\symrefemph#1{\textcolor{cyan}{#1}}
7054 \def\compemph#1{\textcolor{blue}{#1}}
7055 \def\titleemph#1{\textcolor{blue}{#1}}
7056 \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.

```

7057 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
7058 \def\smalltextwarning{
7059   \pgfuseimage{miko@small@dbend}
7060   \xspace
7061 }
7062 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
7063 \newrobustcmd\textwarning{
7064   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
7065   \xspace
7066 }
7067 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
7068 \newrobustcmd\bigtextwarning{
7069   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
7070   \xspace
7071 }

```

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

```

7072 \newrobustcmd\putgraphicsat[3]{
7073   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
7074 }
7075 \newrobustcmd\putat[2]{
7076   \begin{picture}(0,0)\put(#1){#2}\end{picture}
7077 }

```

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`.

```

7078 \bool_if:NT \c__notesslides_sectocframes_bool {
7079   \str_if_eq:VnTF \__notesslidesstopsect{part}{
7080     \newcounter{chapter}\counterwithin*{section}{chapter}
7081   }{
7082     \str_if_eq:VnTF \__notesslidesstopsect{chapter}{
7083       \newcounter{chapter}\counterwithin*{section}{chapter}
7084     }
7085   }
7086 }

```

`\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
7087 \def\part@prefix{}
7088 \@ifpackageloaded{document-structure}{}{
7089   \str_case:VnF \__notesslidesstopsect {
7090     {part}{
7091       \int_set:Nn \l_document_structure_section_level_int {0}
7092       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

7093     \def\part@prefix{\arabic{chapter}.}
7094   }
7095   {chapter}{
7096     \int_set:Nn \l_document_structure_section_level_int {1}
7097     \def\thesection{\arabic{chapter}.\arabic{section}}
7098     \def\part@prefix{\arabic{chapter}.}
7099   }
7100 }{
7101   \int_set:Nn \l_document_structure_section_level_int {2}
7102   \def\part@prefix{}
7103 }
7104 }
7105
7106 \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 `omgroup` environment that chooses the L^AT_EX sectioning macros according to `\section@level`.

sfragment

```

7107 \renewenvironment{sfragment}[2][]{
7108   \_document_structure_omgroup_args:n { #1 }
7109   \int_incr:N \l_document_structure_section_level_int
7110   \bool_if:NT \c__notesslides_sectocframes_bool {
7111     \stepcounter{slide}
7112     \begin{frame}[noframenumbering]
7113     \vfill\Large\centering
7114     \red{
7115       \ifcase\l_document_structure_section_level_int\or
7116         \stepcounter{part}
7117         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
7118         \def\currentsectionlevel{\omdoc@part@kw}
7119       \or
7120         \stepcounter{chapter}
7121         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
7122         \def\currentsectionlevel{\omdoc@chapter@kw}
7123       \or
7124         \stepcounter{section}
7125         \def\_notesslideslabel{\part@prefix\arabic{section}}
7126         \def\currentsectionlevel{\omdoc@section@kw}
7127       \or
7128         \stepcounter{subsection}
7129         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
7130         \def\currentsectionlevel{\omdoc@subsection@kw}
7131       \or
7132         \stepcounter{subsubsection}
7133         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7134         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7135       \or
7136         \stepcounter{paragraph}
7137         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7138         \def\currentsectionlevel{\omdoc@paragraph@kw}
7139       \else
7140         \def\_notesslideslabel{}

```

```

7141         \def\currentsectionlevel{\omdoc@paragraph@kw}
7142         \fi% end ifcase
7143         \_notesslideslabel%\sref@label@id\_notesslideslabel
7144         \quad #2%
7145     }%
7146     \vfill%
7147     \end{frame}%
7148 }
7149 \str_if_empty:NF \l__document_structure_omgroup_id_str {
7150     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7151 }
7152 }{}
7153 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

7154 \def\inserttheorembodyfont{\normalfont}
7155 %\bool_if:NF \c__notesslides_notes_bool {
7156 % \defbeamertemplate{theorem begin}{miko}
7157 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7158 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7159 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7160 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

7161 % \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.

```

7162 % \expandafter\def\csname Parent2\endcsname{}
7163 %}
7164
7165 \AddToHook{begindocument}{% this does not work for some reason
7166     \setbeamertemplate{theorems}[ams style]
7167 }
7168 \bool_if:NT \c__notesslides_notes_bool {
7169     \renewenvironment{columns}[1][{}]{%
7170         \par\noindent%
7171         \begin{minipage}%
7172             \slidewidth\centering\leavevmode%
7173     }{}%
7174     \end{minipage}\par\noindent%
7175 }%
7176 \newsavebox\columnbox%
7177 \renewenvironment<>{column}[2][{}]{%
7178     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7179 }{}%
7180     \end{minipage}\end{lrbox}\usebox\columnbox%
7181 }%
7182 }
7183 \bool_if:NFT \c__notesslides_noproblems_bool {
7184     \newenvironment{problems}{}{}
7185 }{
7186     \excludecomment{problems}
7187 }

```

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.

```

7188 \gdef\printexcursions{}
7189 \newcommand\excursionref[2]{% label, text
7190   \bool_if:NT \c__notesslides_notes_bool {
7191     \begin{sparagraph}[title=Excursion]
7192       #2 \sref[fallback=the appendix]{#1}.
7193     \end{sparagraph}
7194   }
7195 }
7196 \newcommand\activate@excursion[2][]{
7197   \gappto\printexcursions{\inputref{#1}{#2}}
7198 }
7199 \newcommand\excursion[4][]{% repos, label, path, text
7200   \bool_if:NT \c__notesslides_notes_bool {
7201     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7202   }
7203 }

```

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

`\excursiongroup`

```

7204 \keys_define:nn{notesslides / excursiongroup }{
7205   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7206   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7207   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7208 }
7209 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7210   \tl_clear:N \l__notesslides_excursion_intro_tl
7211   \str_clear:N \l__notesslides_excursion_id_str
7212   \str_clear:N \l__notesslides_excursion_mhrepos_str
7213   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7214 }
7215 \newcommand\excursiongroup[1][]{
7216   \__notesslides_excursion_args:n{ #1 }
7217   \ifdefempty\printexcursions{}% only if there are excursions
7218   {\begin{note}
7219     \begin{sfragment}[#1]{Excursions}%
7220     \ifdefempty\l__notesslides_excursion_intro_tl{\
7221       \inputref[\l__notesslides_excursion_mhrepos_str]{
7222         \l__notesslides_excursion_intro_tl
7223       }
7224     }
7225     \printexcursions%
7226     \end{sfragment}
7227   \end{note}}
7228 }
7229 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7230 \</package>

```

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

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.

```
7231 <*package>
7232 <@@=problems>
7233 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7234 \RequirePackage{l3keys2e,stex}
7235
7236 \keys_define:nn { problem / pkg }{
7237   notes      .default:n    = { true },
7238   notes      .bool_set:N   = \c__problems_notes_bool,
7239   gnotes     .default:n    = { true },
7240   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7241   hints      .default:n    = { true },
7242   hints      .bool_set:N   = \c__problems_hints_bool,
7243   solutions  .default:n    = { true },
7244   solutions  .bool_set:N   = \c__problems_solutions_bool,
7245   pts        .default:n    = { true },
7246   pts        .bool_set:N   = \c__problems_pts_bool,
7247   min        .default:n    = { true },
7248   min        .bool_set:N   = \c__problems_min_bool,
7249   boxed      .default:n    = { true },
7250   boxed      .bool_set:N   = \c__problems_boxed_bool,
7251   unknown    .code:n       = {}
7252 }
7253 \newif\ifsolutions
7254
7255 \ProcessKeysOptions{ problem / pkg }
7256 \bool_if:NTF \c__problems_solutions_bool {
7257   \solutionstrue
7258 }{
7259   \solutionsfalse
7260 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7261 \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.

```
7262 \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.

```
7263 \def\prob@problem@kw{Problem}
7264 \def\prob@solution@kw{Solution}
7265 \def\prob@hint@kw{Hint}
7266 \def\prob@note@kw{Note}
7267 \def\prob@gnote@kw{Grading}
7268 \def\prob@pt@kw{pt}
7269 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7270 \AddToHook{begindocument}{
7271   \ltx@ifpackageloaded{babel}{
7272     \makeatletter
7273     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7274     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7275       \input{problem-ngerman.ldf}
7276     }
7277     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7278       \input{problem-finnish.ldf}
7279     }
7280     \clist_if_in:NnT \l_tmpa_clist {french}{
7281       \input{problem-french.ldf}
7282     }
7283     \clist_if_in:NnT \l_tmpa_clist {russian}{
7284       \input{problem-russian.ldf}
7285     }
7286     \makeatother
7287   }{ }
7288 }
```

39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
7289 \keys_define:nn{ problem / problem }{
7290   id      .str_set:x:N = \l__problems_prob_id_str,
7291   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7292   min     .tl_set:N    = \l__problems_prob_min_tl,
7293   title   .tl_set:N    = \l__problems_prob_title_tl,
7294   type    .tl_set:N    = \l__problems_prob_type_tl,
7295   refnum  .int_set:N   = \l__problems_prob_refnum_int
7296 }
7297 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

7298 \str_clear:N \l__problems_prob_id_str
7299 \tl_clear:N \l__problems_prob_pts_tl
7300 \tl_clear:N \l__problems_prob_min_tl
7301 \tl_clear:N \l__problems_prob_title_tl
7302 \tl_clear:N \l__problems_prob_type_tl
7303 \int_zero_new:N \l__problems_prob_refnum_int
7304 \keys_set:nn { problem / problem }{ #1 }
7305 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7306   \let\l__problems_prob_refnum_int\undefined
7307 }
7308 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7309 \newcounter{problem}
7310 \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.

```

7311 \newcommand\prob@label[1]{#1}

```

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

`\prob@number` We consolidate the problem number into a reusable internal macro

```

7312 \newcommand\prob@number{
7313   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7314     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7315   }{
7316     \int_if_exist:NTF \l__problems_prob_refnum_int {
7317       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7318     }{
7319       \prob@label\theproblem
7320     }
7321   }
7322 }

```

(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.

```

7323 \newcommand\prob@title[3]{%
7324   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7325     #2 \l__problems_inclprob_title_tl #3
7326   }{
7327     \tl_if_exist:NTF \l__problems_prob_title_tl {
7328       #2 \l__problems_prob_title_tl #3
7329     }{
7330       #1
7331     }
7332   }
7333 }

```


(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.

```

7334 \def\prob@heading{
7335   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7336   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
7337 }

```

(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`

```

7338 \newenvironment{sproblem}[1][]{
7339   \__problems_prob_args:n{#1}%\sref@target%
7340   \@in@omtexttrue% we are in a statement (for inline definitions)
7341   \stepcounter{problem}\record@problem
7342   \def\current@section@level{\prob@problem@kw}
7343   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7344     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7345   }{
7346     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7347   }
7348   \str_if_exist:NTF \l__problems_inclprob_id_str {
7349     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7350   }{
7351     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7352   }
7353
7354
7355   \clist_set:No \l_tmpa_clist \sproblemtype
7356   \tl_clear:N \l_tmpa_tl
7357   \clist_map_inline:Nn \l_tmpa_clist {
7358     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
7359       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
7360     }
7361   }
7362   \tl_if_empty:NTF \l_tmpa_tl {
7363     \__problems_sproblem_start:
7364   }{
7365     \l_tmpa_tl
7366   }
7367   \stex_ref_new_doc_target:n \sproblemid
7368 }{
7369   \clist_set:No \l_tmpa_clist \sproblemtype
7370   \tl_clear:N \l_tmpa_tl
7371   \clist_map_inline:Nn \l_tmpa_clist {
7372     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
7373       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
7374     }

```

```

7375 }
7376 \tl_if_empty:NTF \l_tmpa_tl {
7377   \__problems_sproblem_end:
7378 }{
7379   \l_tmpa_tl
7380 }
7381
7382
7383 \smallskip
7384 }
7385
7386
7387 \cs_new_protected:Nn \__problems_sproblem_start: {
7388   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7389 }
7390 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7391
7392 \newcommand\stexpatchproblem[3][] {
7393   \str_set:Nx \l_tmpa_str{ #1 }
7394   \str_if_empty:NTF \l_tmpa_str {
7395     \tl_set:Nn \__problems_sproblem_start: { #2 }
7396     \tl_set:Nn \__problems_sproblem_end: { #3 }
7397   }{
7398     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7399     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7400   }
7401 }
7402
7403
7404 \bool_if:NT \c__problems_boxed_bool {
7405   \surroundwithmdframed{problem}
7406 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

7407 \def\record@problem{
7408   \protected@write\@auxout{}
7409   {
7410     \string\@problem{\prob@number}
7411     {
7412       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7413         \l__problems_inclprob_pts_tl
7414       }{
7415         \l__problems_prob_pts_tl
7416       }
7417     }%
7418     {
7419       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7420         \l__problems_inclprob_min_tl
7421       }{
7422         \l__problems_prob_min_tl
7423       }
7424     }
7425   }
7426 }

```

(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).

```
7427 \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.

```
7428 \keys_define:nn { problem / solution }{
7429   id               .str_set_x:N = \l__problems_solution_id_str ,
7430   for              .tl_set:N    = \l__problems_solution_for_tl ,
7431   height           .dim_set:N   = \l__problems_solution_height_dim ,
7432   creators         .clist_set:N = \l__problems_solution_creators_clist ,
7433   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
7434   srccite          .tl_set:N    = \l__problems_solution_srccite_tl
7435 }
7436 \cs_new_protected:Nn \__problems_solution_args:n {
7437   \str_clear:N \l__problems_solution_id_str
7438   \tl_clear:N \l__problems_solution_for_tl
7439   \tl_clear:N \l__problems_solution_srccite_tl
7440   \clist_clear:N \l__problems_solution_creators_clist
7441   \clist_clear:N \l__problems_solution_contributors_clist
7442   \dim_zero:N \l__problems_solution_height_dim
7443   \keys_set:nn { problem / solution }{ #1 }
7444 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7445 \newcommand\@startsolution[1][{}]{
7446   \__problems_solution_args:n { #1 }
7447   \@in@omtexttrue% we are in a statement.
7448   \bool_if:NF \c__problems_boxed_bool { \hrule }
7449   \smallskip\noindent
7450   {\textbf\prob@solution@kw :\enspace}
7451   \begin{small}
7452   \def\current@section@level{\prob@solution@kw}
7453   \ignorespacesandpars
7454 }
```

\startsolutions for the **\startsolutions** macro we use the **\specialcomment** macro from the **comment** package. Note that we use the **\@startsolution** macro in the start codes, that parses the optional argument.

```
7455 \newcommand\startsolutions{
7456   \specialcomment{solution}{\@startsolution}{
7457     \bool_if:NF \c__problems_boxed_bool {
7458       \hrule\medskip
7459     }
7460     \end{small}%
7461   }
7462   \bool_if:NT \c__problems_boxed_bool {
7463     \surroundwithmdframed{solution}
7464   }
7465 }
```

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

\stopsolutions

```
7466 \newcommand\stopsolutions{\excludecomment{solution}}
```

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

so it only remains to start/stop solutions depending on what option was specified.

```
7467 \ifsolutions
7468 \startsolutions
7469 \else
7470 \stopsolutions
7471 \fi
```

exnote

```
7472 \bool_if:NTF \c__problems_notes_bool {
7473 \newenvironment{exnote}[1][]{
7474 \par\smallskip\hrule\smallskip
7475 \noindent\textbf{\prob@note@kw : }\small
7476 }{
7477 \smallskip\hrule
7478 }
7479 }{
7480 \excludecomment{exnote}
7481 }
```

hint

```
7482 \bool_if:NTF \c__problems_notes_bool {
7483 \newenvironment{hint}[1][]{
7484 \par\smallskip\hrule\smallskip
7485 \noindent\textbf{\prob@hint@kw :~ }\small
7486 }{
7487 \smallskip\hrule
7488 }
7489 \newenvironment{exhint}[1][]{
7490 \par\smallskip\hrule\smallskip
7491 \noindent\textbf{\prob@hint@kw :~ }\small
7492 }{
7493 \smallskip\hrule
7494 }
7495 }{
7496 \excludecomment{hint}
7497 \excludecomment{exhint}
7498 }
```

gnote

```
7499 \bool_if:NTF \c__problems_notes_bool {
7500 \newenvironment{gnote}[1][]{
7501 \par\smallskip\hrule\smallskip
7502 \noindent\textbf{\prob@gnote@kw : }\small
7503 }{
7504 \smallskip\hrule
7505 }
7506 }{
7507 \excludecomment{gnote}
7508 }
```

39.3 Multiple Choice Blocks

EdN:17

mcb 17

```
7509 \newenvironment{mcb}{
7510   \begin{enumerate}
7511 }{
7512   \end{enumerate}
7513 }
```

we define the keys for the mcc macro

```
7514 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7515   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7516     \bool_set_true:N #1
7517   }{
7518     \bool_set_false:N #1
7519   }
7520 }
7521 \keys_define:nn { problem / mcc }{
7522   id          .str_set_x:N = \l__problems_mcc_id_str ,
7523   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7524   T           .default:n    = { true } ,
7525   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7526   F           .default:n    = { true } ,
7527   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7528   Ttext       .code:n       = {
7529     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7530   } ,
7531   Ftext       .code:n       = {
7532     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7533   }
7534 }
7535 \cs_new_protected:Nn \l__problems_mcc_args:n {
7536   \str_clear:N \l__problems_mcc_id_str
7537   \tl_clear:N \l__problems_mcc_feedback_tl
7538   \bool_set_true:N \l__problems_mcc_t_bool
7539   \bool_set_true:N \l__problems_mcc_f_bool
7540   \bool_set_true:N \l__problems_mcc_Ttext_bool
7541   \bool_set_false:N \l__problems_mcc_Ftext_bool
7542   \keys_set:nn { problem / mcc }{ #1 }
7543 }
```

\mcc

```
7544 \newcommand\mcc[2][] {
7545   \l__problems_mcc_args:n{ #1 }
7546   \item #2
7547   \ifsolutions
7548     \l
7549     \bool_if:NT \l__problems_mcc_t_bool {
7550       % TODO!
7551       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
7552     }
7553     \bool_if:NT \l__problems_mcc_f_bool {
```

¹⁷EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7554      % TODO!
7555      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7556    }
7557    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7558      !
7559    }{
7560      \l__problems_mcc_feedback_tl
7561    }
7562    \fi
7563  } %solutions

```

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

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.

```

7564
7565 \keys_define:nn{ problem / inclproblem }{
7566   id      .str_set:N = \l__problems_inclprob_id_str,
7567   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7568   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7569   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7570   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7571   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7572   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7573 }
7574 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7575   \str_clear:N \l__problems_prob_id_str
7576   \tl_clear:N \l__problems_inclprob_pts_tl
7577   \tl_clear:N \l__problems_inclprob_min_tl
7578   \tl_clear:N \l__problems_inclprob_title_tl
7579   \tl_clear:N \l__problems_inclprob_type_tl
7580   \int_zero_new:N \l__problems_inclprob_refnum_int
7581   \str_clear:N \l__problems_inclprob_mhrepos_str
7582   \keys_set:nn { problem / inclproblem }{ #1 }
7583   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7584     \let\l__problems_inclprob_pts_tl\undefined
7585   }
7586   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7587     \let\l__problems_inclprob_min_tl\undefined
7588   }
7589   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7590     \let\l__problems_inclprob_title_tl\undefined
7591   }
7592   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7593     \let\l__problems_inclprob_type_tl\undefined
7594   }
7595   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7596     \let\l__problems_inclprob_refnum_int\undefined
7597   }
7598 }

```

```

7599
7600 \cs_new_protected:Nn \__problems_inclprob_clear: {
7601   \let\l__problems_inclprob_id_str\undefined
7602   \let\l__problems_inclprob_pts_tl\undefined
7603   \let\l__problems_inclprob_min_tl\undefined
7604   \let\l__problems_inclprob_title_tl\undefined
7605   \let\l__problems_inclprob_type_tl\undefined
7606   \let\l__problems_inclprob_refnum_int\undefined
7607   \let\l__problems_inclprob_mhrepos_str\undefined
7608 }
7609 \__problems_inclprob_clear:
7610
7611 \newcommand\includeproblem[2][ ]{
7612   \__problems_inclprob_args:n{ #1 }
7613   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7614     \input{#2}
7615   }{
7616     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7617       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7618     }
7619   }
7620   \__problems_inclprob_clear:
7621 }

```

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

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.

```

7622 \AddToHook{enddocument}{
7623   \bool_if:NT \c__problems_pts_bool {
7624     \message{Total:~\arabic{pts}~points}
7625   }
7626   \bool_if:NT \c__problems_min_bool {
7627     \message{Total:~\arabic{min}~minutes}
7628   }
7629 }

```

The margin pars are reader-visible, so we need to translate

```

7630 \def\pts#1{
7631   \bool_if:NT \c__problems_pts_bool {
7632     \marginpar{#1~\prob@pt@kw}
7633   }
7634 }
7635 \def\min#1{
7636   \bool_if:NT \c__problems_min_bool {
7637     \marginpar{#1~\prob@min@kw}
7638   }
7639 }

```

`\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.

```

7640 \newcounter{pts}
7641 \def\show@pts{
7642   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7643     \bool_if:NT \c__problems_pts_bool {
7644       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7645       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7646     }
7647   }{
7648     \tl_if_exist:NT \l__problems_prob_pts_tl {
7649       \bool_if:NT \c__problems_pts_bool {
7650         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7651         \addtocounter{pts}{\l__problems_prob_pts_tl}
7652       }
7653     }
7654   }
7655 }

```

(End definition for `\show@pts`. This function is documented on page ??.)
and now the same for the minutes

`\show@min`

```

7656 \newcounter{min}
7657 \def\show@min{
7658   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7659     \bool_if:NT \c__problems_min_bool {
7660       \marginpar{\l__problems_inclprob_min_tl\ min}
7661       \addtocounter{min}{\l__problems_inclprob_min_tl}
7662     }
7663   }{
7664     \tl_if_exist:NT \l__problems_prob_min_tl {
7665       \bool_if:NT \c__problems_min_bool {
7666         \marginpar{\l__problems_prob_min_tl\ min}
7667         \addtocounter{min}{\l__problems_prob_min_tl}
7668       }
7669     }
7670   }
7671 }
7672 \</package>

```

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

Chapter 40

Implementation: The hwexam Class

The functionality is spread over the `hwexam` class and package. The class provides the `document` environment and pre-loads some convenience packages, whereas the package provides the concrete functionality.

40.1 Class Options

To initialize the `hwexam` class, we declare and process the necessary options by passing them to the respective packages and classes they come from.

```
7673 <@@=hwexam>
7674 <*cls>
7675 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7676 \RequirePackage{l3keys2e}
7677 \DeclareOption*{
7678   \PassOptionsToClass{\CurrentOption}{document-structure}
7679   \PassOptionsToPackage{\CurrentOption}{stex}
7680   \PassOptionsToPackage{\CurrentOption}{hwexam}
7681   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7682 }
7683 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
7684 \LoadClass{document-structure}
7685 \RequirePackage{stex}
7686 \RequirePackage{hwexam}
7687 \RequirePackage{tikzinput}
7688 \RequirePackage{graphicx}
7689 \RequirePackage{a4wide}
7690 \RequirePackage{amssymb}
7691 \RequirePackage{amstext}
7692 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors

```

7693 \newcommand\assig@default@type{\hwexam@assignment@kw}
7694 \def\document@hwexamtype{\assig@default@type}
7695 <@@=document_structure>
7696 \keys_define:nn { document-structure / document }{
7697 id .str_set_x:N = \c_document_structure_document_id_str,
7698 hwexamtype .tl_set:N = \document@hwexamtype
7699 }
7700 <@@=hwexam>
7701 </cls>

```

Chapter 41

Implementation: The hwexam Package

41.1 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.

```
7702 \langle *package \rangle
7703 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7704 \RequirePackage{13keys2e}
7705
7706 \newif\iftest\testfalse
7707 \DeclareOption{test}{\testtrue}
7708 \newif\ifmultiple\multiplefalse
7709 \DeclareOption{multiple}{\multipletrue}
7710 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7711 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7712 \RequirePackage{keyval}[1997/11/10]
7713 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7714 \newcommand\hwexam@assignment@kw{Assignment}
7715 \newcommand\hwexam@given@kw{Given}
7716 \newcommand\hwexam@due@kw{Due}
7717 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7718 blank~for~extra~space}
7719 \def\hwexam@minutes@kw{minutes}
7720 \newcommand\correction@probs@kw{prob.}
7721 \newcommand\correction@pts@kw{total}
7722 \newcommand\correction@reached@kw{reached}
7723 \newcommand\correction@sum@kw{Sum}
7724 \newcommand\correction@grade@kw{grade}
7725 \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

```

7726 \AddToHook{begindocument}{
7727 \ltx@ifpackageloaded{babel}{
7728 \makeatletter
7729 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7730 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7731 \input{hwexam-ngerman.ldf}
7732 }
7733 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7734 \input{hwexam-finnish.ldf}
7735 }
7736 \clist_if_in:NnT \l_tmpa_clist {french}{
7737 \input{hwexam-french.ldf}
7738 }
7739 \clist_if_in:NnT \l_tmpa_clist {russian}{
7740 \input{hwexam-russian.ldf}
7741 }
7742 \makeatother
7743 }{}
7744 }
7745

```

41.2 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.

```

7746 \newcounter{assignment}
7747 \numberproblemsin{assignment}
7748 \renewcommand\prob@label[1]{\assignment@number.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

7749 \keys_define:nn { hwexam / assignment } {
7750 id .str_set:N = \l__hwexam_assign_id_str,
7751 number .int_set:N = \l__hwexam_assign_number_int,
7752 title .tl_set:N = \l__hwexam_assign_title_tl,
7753 type .tl_set:N = \l__hwexam_assign_type_tl,
7754 given .tl_set:N = \l__hwexam_assign_given_tl,
7755 due .tl_set:N = \l__hwexam_assign_due_tl,
7756 loadmodules .code:n = {
7757 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7758 }
7759 }
7760 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7761 \str_clear:N \l__hwexam_assign_id_str
7762 \int_set:Nn \l__hwexam_assign_number_int {-1}
7763 \tl_clear:N \l__hwexam_assign_title_tl
7764 \tl_clear:N \l__hwexam_assign_type_tl
7765 \tl_clear:N \l__hwexam_assign_given_tl
7766 \tl_clear:N \l__hwexam_assign_due_tl
7767 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7768 \keys_set:nn { hwexam / assignment }{ #1 }
7769 }

```

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.

```

7770 \newcommand\given@due[2]{
7771 \bool_lazy_all:nF {
7772 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7773 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7774 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7775 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7776 }{ #1 }
7777
7778 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7779 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7780 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7781 }
7782 }{
7783 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7784 }
7785
7786 \bool_lazy_or:nnF {
7787 \bool_lazy_and_p:nn {
7788 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7789 }{
7790 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7791 }
7792 }{
7793 \bool_lazy_and_p:nn {
7794 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7795 }{
7796 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7797 }
7798 }{ ,~ }
7799
7800 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7801 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7802 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7803 }
7804 }{
7805 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7806 }
7807
7808 \bool_lazy_all:nF {
7809 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7810 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7811 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7812 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7813 }{ #2 }
7814 }

```

`\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.

```

7815 \newcommand\assignment@title[3]{
7816 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7817 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7818 #1
7819 }{
7820 #2\l__hwexam_assign_title_tl#3
7821 }
7822 }{
7823 #2\l__hwexam_inclasssign_title_tl#3
7824 }
7825 }

```

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

`\assignment@number` Like `\assignment@title` only for the number, and no around part.

```

7826 \newcommand\assignment@number{
7827 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7828 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7829 \arabic{assignment}
7830 } {
7831 \int_use:N \l__hwexam_assign_number_int
7832 }
7833 }{
7834 \int_use:N \l__hwexam_inclasssign_number_int
7835 }
7836 }

```

(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.

```

7837 \newenvironment{assignment}[1][ ]{
7838 \__hwexam_assignment_args:n { #1 }
7839 %\sref@target
7840 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7841 \global\stepcounter{assignment}
7842 }{
7843 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7844 }
7845 \setcounter{problem}{0}
7846 \def\current@section@level{\document@hwexamtype}
7847 %\sref@label@id{\document@hwexamtype \thesection}
7848 \begin{@assignment}
7849 }{
7850 \end{@assignment}
7851 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

7852 \def\ass@title{
7853 \protect\document@hwexamtype~\arabic{assignment}
7854 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7855 }
7856 \ifmultiple
7857 \newenvironment{@assignment}{
7858 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7859 \begin{sfragment}[loadmodules]{\ass@title}
7860 }{
7861 \begin{sfragment}{\ass@title}
7862 }
7863 }{
7864 \end{sfragment}
7865 }

```

for the single-page case we make a title block from the same components.

```

7866 \else
7867 \newenvironment{@assignment}{
7868 \begin{center}\bf
7869 \Large@title\strut\
7870 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7871 \large\given@due{--;\}{}\{;\}{}
7872 \end{center}
7873 }{}
7874 \fi% multiple

```

41.3 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.

```

7875 \keys_define:nn { hwexam / inclassignment } {
7876 %id .str_set_x:N = \l__hwexam_assign_id_str,
7877 number .int_set:N = \l__hwexam_inclassign_number_int,
7878 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7879 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7880 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7881 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7882 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7883 }
7884 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7885 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7886 \tl_clear:N \l__hwexam_inclassign_title_tl
7887 \tl_clear:N \l__hwexam_inclassign_type_tl
7888 \tl_clear:N \l__hwexam_inclassign_given_tl
7889 \tl_clear:N \l__hwexam_inclassign_due_tl
7890 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7891 \keys_set:nn { hwexam / inclassignment }{ #1 }
7892 }
7893 \__hwexam_inclassignment_args:n {}
7894
7895 \newcommand\inputassignment[2][{}]{

```

```

7896 \_hwexam_inclassnment_args:n { #1 }
7897 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7898 \input{#2}
7899 }{
7900 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7901 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7902 }
7903 }
7904 \_hwexam_inclassnment_args:n {}
7905 }
7906 \newcommand\includeassignment[2][]{
7907 \newpage
7908 \inputassignment[#1]{#2}
7909 }

```

(End definition for \in*assignment. This function is documented on page ??.)

41.4 Typesetting Exams

\quizheading

```

7910 \ExplSyntaxOff
7911 \newcommand\quizheading[1]{%
7912 \def\@tas{#1}%
7913 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7914 \ifx\@tas\@empty\else%
7915 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7916 \fi%
7917 }
7918 \ExplSyntaxOn

```

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

\testheading

```

7919
7920 \def\hwexamheader{\input{hwexam-default.header}}
7921
7922 \def\hwexamminutes{
7923 \tl_if_empty:NTF \testheading@duration {
7924 {\testheading@min}~\hwexam@minutes@kw
7925 }{
7926 \testheading@duration
7927 }
7928 }
7929
7930 \keys_define:nn { hwexam / testheading } {
7931 min .tl_set:N = \testheading@min,
7932 duration .tl_set:N = \testheading@duration,
7933 reqpts .tl_set:N = \testheading@reqpts,
7934 tools .tl_set:N = \testheading@tools
7935 }
7936 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7937 \tl_clear:N \testheading@min
7938 \tl_clear:N \testheading@duration

```



```

7939 \tl_clear:N \testheading@reqpts
7940 \tl_clear:N \testheading@tools
7941 \keys_set:nn { hwexam / testheading }{ #1 }
7942 }
7943 \newenvironment{testheading}[1][]{
7944   \_hwexam_testheading_args:n{ #1 }
7945   \newcount\check@time\check@time=\testheading@min
7946   \advance\check@time by -\theassignment@totalmin
7947   \newif\if@bonuspoints
7948   \tl_if_empty:NTF \testheading@reqpts {
7949     \@bonuspointsfalse
7950   }{
7951     \newcount\bonus@pts
7952     \bonus@pts=\theassignment@totalpts
7953     \advance\bonus@pts by -\testheading@reqpts
7954     \edef\bonus@pts{\the\bonus@pts}
7955     \@bonuspointstrue
7956   }
7957   \edef\check@time{\the\check@time}
7958
7959   \makeatletter\hwexamheader\makeatother
7960 }{
7961   \newpage
7962 }

```

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

\testspace

```

7963 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7964 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7965 \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.

```

7966 <@=problems>
7967 \renewcommand\@problem[3]{
7968   \stepcounter{assignment@probs}
7969   \def\__problemspts{#2}
7970   \ifx\__problemspts\@empty\else
7971     \addtocounter{assignment@totalpts}{#2}
7972   \fi
7973   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7974   \xdef\correction@probs{\correction@probs & #1}%
7975   \xdef\correction@pts{\correction@pts & #2}
7976   \xdef\correction@reached{\correction@reached &}

```

```

7977 }
7978 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

7979 \newcounter{assignment@probs}
7980 \newcounter{assignment@totalpts}
7981 \newcounter{assignment@totalmin}
7982 \def\correction@probs{\correction@probs@kw}
7983 \def\correction@pts{\correction@pts@kw}
7984 \def\correction@reached{\correction@reached@kw}
7985 \stepcounter{assignment@probs}
7986 \newcommand\correction@table{
7987 \resizebox{\textwidth}{!}{%
7988 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7989 &\multicolumn{\theassignment@probs}{c|}|%|
7990 {\footnotesize\correction@forgrading@kw} &\\ \hline
7991 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7992 \correction@pts & \theassignment@totalpts & \\ \hline
7993 \correction@reached & & \[.7cm]\hline
7994 \end{tabular}}
7995 </package>

```

(End definition for \correction@table. This function is documented on page ??.)

41.5 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}

```