

The sTeX3 Package *

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

2022-03-07

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 2022-03-07)

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	102
25.1	Generic Path Handling	102
25.2	PWD and kpsewhich	104
25.3	File Hooks and Tracking	105
25.4	MathHub Repositories	106
25.5	Using Content in Archives	110
26	gTeX-References Implementation	115
26.1	Document URIs and URLs	115
26.2	Setting Reference Targets	117
26.3	Using References	119
27	gTeX-Modules Implementation	122
27.1	The smodule environment	126
27.2	Invoking modules	131
28	gTeX-Module Inheritance Implementation	133
28.1	SMS Mode	133
28.2	Inheritance	136

29	STEX-Symbols Implementation	141
29.1	Symbol Declarations	141
29.2	Notations	148
29.3	Variables	157
30	STEX-Terms Implementation	164
30.1	Symbol Invocations	164
30.2	Terms	171
30.3	Notation Components	175
30.4	Variables	177
30.5	Sequences	179
31	STEX-Structural Features Implementation	180
31.1	Imports with modification	181
31.2	The feature environment	187
31.3	Structure	188
32	STEX-Statements Implementation	197
32.1	Definitions	197
32.2	Assertions	202
32.3	Examples	205
32.4	Logical Paragraphs	208
33	The Implementation	213
33.1	Package Options	213
33.2	Proofs	213
33.3	Justifications	224
34	STEX-Others Implementation	226
35	STEX-Metatheory Implementation	227
36	Tikzinput Implementation	230
37	document-structure.sty Implementation	232
37.1	The document-structure Class	232
37.2	Class Options	232
37.3	Beefing up the document environment	233
37.4	Implementation: document-structure Package	233
37.5	Package Options	233
37.6	Document Structure	235
37.7	Front and Backmatter	238
37.8	Global Variables	240

38 NotesSlides – Implementation	241
38.1 Class and Package Options	241
38.2 Notes and Slides	243
38.3 Header and Footer Lines	247
38.4 Frame Images	248
38.5 Colors and Highlighting	249
38.6 Sectioning	250
38.7 Excursions	253
39 The Implementation	254
39.1 Package Options	254
39.2 Problems and Solutions	255
39.3 Multiple Choice Blocks	261
39.4 Including Problems	262
39.5 Reporting Metadata	263
40 Implementation: The hwexam Class	265
40.1 Class Options	265
41 Implementation: The hwexam Package	267
41.1 Package Options	267
41.2 Assignments	268
41.3 Including Assignments	271
41.4 Typesetting Exams	272
41.5 Leftovers	274

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     \[\defeq{\geometricSeries}{\definiens{
14       \infinitesum{\svar{n}}{1}{
15         \realdivide[frac]{1}{
16           \realpower{2}{\svar{n}}
17         }
18       }}
19     \].\]
20   \end{sdefinition}
21
22   \begin{sassertion}[name=geometricSeriesConverges,type=theorem]
23     The \symname{geometricSeries} \symname{converges} towards $1$.
24   \end{sassertion}
25 \end{smodule}
26 \end{document}

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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 \LaTeX 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 `<namespace>?<module-name>`.

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 \hookrightarrow `<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

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

`\comp`

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`

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\sim\#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: $\forall a <_S b <_{Sc} c <_{Se} d <_{Se} t$

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.

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

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

In the example above:



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

3.3.6 Variables

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

Variables are different – variables are allowed everywhere, are not exported when the current module (if one exists) is imported (via `\importmodule` or `\usemodule`) and (also unlike symbol declarations) “disappear” at the end of the current T_EX 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{varm}[name=m,type=\Nat]{\comp{m}}
2 \varseq{seqa}[
3   name=a,
4   args=2,
5   type=\Nat,
6 ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}
7
8 \seqa! and \addition{\seqa}
```

Output:

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

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^{\varm}}
6 ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}
7
8 \seqa! and \addition{\seqa}
```

Output:

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

3.4 Module Inheritance and Structures

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 `lcm(a,b)` in *english*, but is called *kleinstes gemeinsames Vielfaches* in *german* and consequently denoted as `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   $\lcm{a,b}$ 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 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 .

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 $\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}$
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 $\Int$...

```

Output:

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

```

Output:

Test: $a \cdot c \cdot c$

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

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

`sdefinition` (and `sparagraph` with `type=symdoc`) introduce three new macros: `definiendum` behaves like `symref` (and `definame` like `symname`), but highlights the references 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).

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

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 is...

Now the **mathstructure** 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, $\text{\texttt{S}\TeX}$ 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 `gTEX` 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 `gTEX` 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 $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

<code>\stex_filestack_push:n</code>	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> <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> <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> <code>\clstinputmhlisting</code> <hr/>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 10

STEX-References

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

10.1 Macros and Environments

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

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 Using References

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

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

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

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

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

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

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

Chapter 11

STEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

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

11.1.1 The `smodule` environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`\stex_activate_module:n`

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

Chapter 12

STEX-Module Inheritance

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

12.1 Macros and Environments

12.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

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

`\g_stex_smsmode_allowedmacros_escape_tl`

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

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

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

Tests whether SMS mode is currently active.

<code>\stex_file_in_smsmode:nn</code>	<code>\stex_in_smsmode:nn {<filename>} {<code>}</code>
---------------------------------------	--

Executes `<code>` in SMS mode, followed by the content of `<filename>`. `<code>` 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.

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

12.1.2 Imports and Inheritance

<code>\importmodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
----------------------------	---

Imports a module by reading it from a file and “activating” it. `STEX` determines the module and its containing file by passing its arguments on to `\stex_import_module_path:nn`.

<code>\usemodule</code>	<code>\importmodule[<archive-ID>]{<module-path>}</code>
-------------------------	---

Like `\importmodule`, but does not export its contents; i.e. including the current module will not activate the used module

\stex_import_module_uri:nn

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

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

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

<hr/> <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 $\S\TeX$ collection, a version of \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in \LaTeX . This includes a simple structure sharing mechanism for $\S\TeX$ 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 $\S\TeX$ sources, or after translation.

20.1 Introduction

$\S\TeX$ is a version of \TeX / \LaTeX that allows to markup \TeX / \LaTeX documents semantically without leaving the document format, essentially turning \TeX / \LaTeX 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 $\S\TeX$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\S\TeX$ 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 \TeX 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 L ^A T _E XML transformation.
	The structure of the document is given by the <code>omgroup</code> environment just like in OM-
	DOC. In the L ^A T _E X 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.barderv,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 \LaTeX 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> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | 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 `noframenumbering` option so that the frame numbering is kept in sync between the slides and the course notes.

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

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.

`\setslidelogo`
`\setsource`
`\setlicensing`

`\frameimage`

`\mhframeimage`

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

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 — 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` 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.

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

ST_EX -Basics Implementation

24.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

24.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2022/03/03}{3.1.0}{sTeX package}
21
22 %\RequirePackage{morewrites}
23 %\RequirePackage{amsmath}
24
25 Package options:
26 \keys_define:nn { stex } {
```

```

26 debug      .clist_set:N = \c_stex_debug_clist ,
27 lang       .clist_set:N = \c_stex_languages_clist ,
28 mathhub    .tl_set_x:N  = \mathhub ,
29 sms        .bool_set:N  = \c_stex_persist_mode_bool ,
30 image      .bool_set:N  = \c_tikzinput_image_bool,
31 unknown    .code:n      = {}
32 }
33 \ProcessKeysOptions { stex }

```

\stex The \TeX logo:

\sTeX

```

34 \protected\def\stex{
35   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
36 }
37 \let\sTeX\stex

```

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

24.3 Messages and logging

```

38 <@@=stex_log>
    Warnings and error messages
39 \msg_new:nnn{stex}{error/unknownlanguage}{
40   Unknown~language:~#1
41 }
42 \msg_new:nnn{stex}{warning/nomathhub}{
43   MATHHUB~system~variable~not~found~and~no~
44   \detokenize{\mathhub}~value~set!
45 }
46 \msg_new:nnn{stex}{error/deactivated-macro}{
47   The~\detokenize{#1}~command~is~only~allowed~in~#2!
48 }

```

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

```

49 \cs_new_protected:Nn \stex_debug:nn {
50   \clist_if_in:NnTF \c_stex_debug_clist { all } {
51     \msg_set:nnn{stex}{debug / #1}{
52       \\Debug~#1:~#2\\
53     }
54     \msg_none:nn{stex}{debug / #1}
55   }{
56     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
57       \msg_set:nnn{stex}{debug / #1}{
58         \\Debug~#1:~#2\\
59       }
60       \msg_none:nn{stex}{debug / #1}
61     }
62   }
63 }

```

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

Redirecting messages:

```

64 \clist_if_in:NnTF \c_stex_debug_clist {all} {
65   \msg_redirect_module:nnn{ stex }{ none }{ term }

```

```

66 }{
67   \clist_map_inline:Nn \c_stex_debug_clist {
68     \msg_redirect_name:nnn{ stex }{ debug / ##1 }{ term }
69   }
70 }
71
72 \stex_debug:nn{log}{debug~mode~on}

```

24.4 HTML Annotations

```

73 <@=stex_annotate>
74 \RequirePackage{rustex}

```

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

```

75 \rustex_add_Namespace:nn{stex}{http://kwarc.info/ns/sTeX}

```

Conditionals for L^AT_EXML:

`\if@latexml`

```

76 \ifcsname if@latexml\endcsname\else
77   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
78 \fi

```

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

`\latexml_if_p:`

`\latexml_if:TF`

```

79 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
80   \if@latexml
81     \prg_return_true:
82   \else:
83     \prg_return_false:
84   \fi:
85 }

```

(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.

```

86 \tl_new:N \l__stex_annotate_arg_tl
87 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
88   \rustex_if:TF {
89     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
90   }{-}
91 }

```

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

`__stex_annotate_checkempty:n`

```

92 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
93   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
94   \tl_if_empty:NT \l__stex_annotate_arg_tl {
95     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
96   }
97 }

```

(End definition for `__stex_annotate_checkempty:n`.)


```

\stex_if_do_html_p: Whether to (locally) produce HTML output
\stex_if_do_html:TF
100
101 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
102   \bool_if:nTF \stex_html_do_output_bool
103     \prg_return_true: \prg_return_false:
104 }

```

(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
105 \cs_new_protected:Nn \stex_suppress_html:n {
106   \exp_args:Nne \use:nn {
107     \bool_set_false:N \stex_html_do_output_bool
108     #1
109   }{
110     \stex_if_do_html:T {
111       \bool_set_true:N \stex_html_do_output_bool
112     }
113   }
114 }

```

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

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

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

```

115 \rustex_if:TF{
116   \cs_new_protected:Nn \stex_annotate:nnn {
117     \__stex_annotate_checkempty:n { #3 }
118     \rustex_annotate_HTML:nn {
119       property="stex:#1" ~
120       resource="#2"
121     } {
122       \mode_if_vertical:TF{
123         \tl_use:N \l__stex_annotate_arg_tl\par
124       }{
125         \tl_use:N \l__stex_annotate_arg_tl
126       }
127     }
128   }
129   \cs_new_protected:Nn \stex_annotate_invisible:n {
130     \__stex_annotate_checkempty:n { #1 }
131     \rustex_annotate_HTML:nn {
132       stex:visible="false" ~
133       style:display="none"
134     } {
135       \mode_if_vertical:TF{
136         \tl_use:N \l__stex_annotate_arg_tl\par
137       }{
138         \tl_use:N \l__stex_annotate_arg_tl
139       }

```

```

140     }
141   }
142   \cs_new_protected:Nn \stex_annotate_invisible:nnn {
143     \__stex_annotate_checkempty:n { #3 }
144     \rustex_annotate_HTML:nn {
145       property="stex:#1" ~
146       resource="#2" ~
147       stex:visible="false" ~
148       style:display="none"
149     } {
150       \mode_if_vertical:TF{
151         \tl_use:N \l__stex_annotate_arg_tl\par
152       }{
153         \tl_use:N \l__stex_annotate_arg_tl
154       }
155     }
156   }
157   \NewDocumentEnvironment{stex_annotate_env} { m m } {
158     \par
159     \rustex_annotate_HTML_begin:n {
160       property="stex:#1" ~
161       resource="#2"
162     }
163   }{
164     \par\rustex_annotate_HTML_end:
165   }
166 }{
167   \latexml_if:TF {
168     \cs_new_protected:Nn \stex_annotate:nnn {
169       \__stex_annotate_checkempty:n { #3 }
170       \mode_if_math:TF {
171         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
172           \tl_use:N \l__stex_annotate_arg_tl
173         }
174       }{
175         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
176           \tl_use:N \l__stex_annotate_arg_tl
177         }
178       }
179     }
180     \cs_new_protected:Nn \stex_annotate_invisible:n {
181       \__stex_annotate_checkempty:n { #1 }
182       \mode_if_math:TF {
183         \cs:w latexml@invisible@math\cs_end:{
184           \tl_use:N \l__stex_annotate_arg_tl
185         }
186       } {
187         \cs:w latexml@invisible@text\cs_end:{
188           \tl_use:N \l__stex_annotate_arg_tl
189         }
190       }
191     }
192     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
193       \__stex_annotate_checkempty:n { #3 }

```

```

194     \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
195       \tl_use:N \l__stex_annotate_arg_tl
196     }
197   }
198   \NewDocumentEnvironment{stex_annotate_env} { m m } {
199     \par\begin{latexml@annotateenv}{#1}{#2}
200   }{
201     \par\end{latexml@annotateenv}
202   }
203 }{
204   \cs_new_protected:Nn \stex_annotate:nnn {#3}
205   \cs_new_protected:Nn \stex_annotate_invisible:n {}
206   \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
207   \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
208 }
209 }

```

(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

```

210 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

211 \prop_const_from_keyval:Nn \c_stex_languages_prop {
212   en = english ,
213   de = ngerman ,
214   ar = arabic ,
215   bg = bulgarian ,
216   ru = russian ,
217   fi = finnish ,
218   ro = romanian ,
219   tr = turkish ,
220   fr = french
221 }
222
223 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
224   english = en ,
225   ngerman = de ,
226   arabic = ar ,
227   bulgarian = bg ,
228   russian = ru ,
229   finnish = fi ,
230   romanian = ro ,
231   turkish = tr ,
232   french = fr
233 }
234 % todo: chinese simplified (zhs)
235 %       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:

```

236 \clist_if_empty:NF \c_stex_languages_clist {
237   \clist_clear:N \l_tmpa_clist
238   \clist_map_inline:Nn \c_stex_languages_clist {
239     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
240       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
241     } {
242       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
243     }
244   }
245   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
246   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
247 }

```

24.6 Auxiliary Methods

\stex_deactivate_macro:Nn

```

248 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
249   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
250   \def#1{
251     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
252   }
253 }

```

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

\stex_reactivate_macro:N

```

254 \cs_new_protected:Nn \stex_reactivate_macro:N {
255   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
256 }

```

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

\ignorespacesandpars

```

257 \protected\def\ignorespacesandpars{
258   \begingroup\catcode13=10\relax
259   \@ifnextchar\par{
260     \endgroup\expandafter\ignorespacesandpars\@gobble
261   }{
262     \endgroup
263   }
264 }
265 \</package>

```

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

Chapter 25

STEX -MathHub Implementation

```
266 <*package>
267
268 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
269
270 <@@=stex_path>
271
272 Warnings and error messages
273 \msg_new:nnn{stex}{error/norepository}{
274   No~archive~#1~found~in~#2
275 }
276 \msg_new:nnn{stex}{error/notinarchive}{
277   Not~currently~in~an~archive,~but~\detokenize{#1}~
278   needs~one!
279 }
280 \msg_new:nnn{stex}{error/nofile}{
281   \detokenize{#1}~could~not~find~file~#2
282 }
283 \msg_new:nnn{stex}{error/twofiles}{
284   \detokenize{#1}~found~two~candidates~for~#2
285 }
```

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`

```
284 \cs_new_protected:Nn \stex_path_from_string:Nn {
285   \str_set:Nx \l_tmpa_str { #2 }
286   \str_if_empty:NTF \l_tmpa_str {
287     \seq_clear:N #1
288   }{
289     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
290     \sys_if_platform_windows:T{
291       \seq_clear:N \l_tmpa_tl
```

```

292 \seq_map_inline:Nn #1 {
293   \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
294   \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
295 }
296 \seq_set_eq:NN #1 \l_tmpa_tl
297 }
298 \stex_path_canonicalize:N #1
299 }
300 }
301

```

(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`

```

302 \cs_new_protected:Nn \stex_path_to_string:NN {
303   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
304 }
305
306 \cs_new:Nn \stex_path_to_string:N {
307   \seq_use:Nn #1 /
308 }

```

(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`

```

309 \str_const:Nn \c__stex_path_dot_str {.}
310 \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.

```

311 \cs_new_protected:Nn \stex_path_canonicalize:N {
312   \seq_if_empty:NF #1 {
313     \seq_clear:N \l_tmpa_seq
314     \seq_get_left:NN #1 \l_tmpa_tl
315     \str_if_empty:NT \l_tmpa_tl {
316       \seq_put_right:Nn \l_tmpa_seq {}
317     }
318     \seq_map_inline:Nn #1 {
319       \str_set:Nn \l_tmpa_tl { ##1 }
320       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
321         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
322           \seq_if_empty:NNTF \l_tmpa_seq {
323             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
324               \c__stex_path_up_str
325             }
326           }{
327             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
328             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
329               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
330                 \c__stex_path_up_str
331               }
332             }{

```

```

333         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
334     }
335 }
336 }{
337     \str_if_empty:NF \l_tmpa_tl {
338         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
339     }
340 }
341 }
342 }
343 \seq_gset_eq:NN #1 \l_tmpa_seq
344 }
345 }

```

(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`

```

346 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
347     \seq_if_empty:NTF #1 {
348         \prg_return_false:
349     }{
350         \seq_get_left:NN #1 \l_tmpa_tl
351         \sys_if_platform_windows:TF{
352             \str_if_in:NnTF \l_tmpa_tl {:}{
353                 \prg_return_true:
354             }{
355                 \prg_return_false:
356             }
357         }{
358             \str_if_empty:NTF \l_tmpa_tl {
359                 \prg_return_true:
360             }{
361                 \prg_return_false:
362             }
363         }
364     }
365 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

366 \str_new:N\l_stex_kpsewhich_return_str
367 \cs_new_protected:Nn \stex_kpsewhich:n {
368     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
369     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
370     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
371 }

```

(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`

```

372 \sys_if_platform_windows:TF{
373   \begingroup\escapechar=-1\catcode'\=12
374   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
375   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
376   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
377   }}{
378   \stex_kpsewhich:n{-var-value~PWD}
379   }
380
381 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
382 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
383 \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

```

384 <@@=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

```

385 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

386 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
387 \stex_path_from_string:Nn \c_stex_mainfile_seq
388   \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`

```

389 \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`

```

390 \cs_new_protected:Nn \stex_filestack_push:n {
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
392   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
393     \stex_path_from_string:Nn\g_stex_currentfile_seq{
394       \c_stex_pwd_str/#1
395     }
396   }
397   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
398   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
399 }

```



```

441 }
442 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
443 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
444 }

```

(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

```

445 \cs_new_protected:Nn \_stex_mathhub_do_manifest:n {
446   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
447     \str_set:Nx \l_tmpa_str { #1 }
448     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
449     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
450     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
451     \_stex_mathhub_find_manifest:N \l_tmpa_seq
452     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
453       \msg_error:nnxx{stex}{error/norepository}{#1}{
454         \stex_path_to_string:N \c_stex_mathhub_str
455       }
456     } {
457       \exp_args:No \_stex_mathhub_parse_manifest:n { \l_tmpa_str }
458     }
459   }
460 }

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

461 \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`:

```

462 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
463   \seq_set_eq:NN\l_tmpa_seq #1
464   \bool_set_true:N\l_tmpa_bool
465   \bool_while_do:Nn \l_tmpa_bool {
466     \seq_if_empty:NTF \l_tmpa_seq {
467       \bool_set_false:N\l_tmpa_bool
468     } {
469       \file_if_exist:nTF{
470         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
471       } {
472         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
473         \bool_set_false:N\l_tmpa_bool
474       } {
475         \file_if_exist:nTF{
476           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
477         } {
478           \seq_put_right:Nn\l_tmpa_seq{META-INF}
479           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

480         \bool_set_false:N\l_tmpa_bool
481     }{
482         \file_if_exist:nTF{
483             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
484         }{
485             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
486             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
487             \bool_set_false:N\l_tmpa_bool
488         }{
489             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
490         }
491     }
492 }
493 }
494 }
495 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
496 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

497 \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:

```

498 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
499     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
500     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
501     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
502         \str_set:Nn \l_tmpa_str {##1}
503         \exp_args:NNoo \seq_set_split:Nnn
504             \l_tmpb_seq \c_colon_str \l_tmpa_str
505         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
506             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
507                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
508             }
509             \exp_args:No \str_case:nnTF \l_tmpa_tl {
510                 {id} {
511                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512                         { id } \l_tmpb_tl
513                 }
514                 {narration-base} {
515                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516                         { narr } \l_tmpb_tl
517                 }
518                 {url-base} {
519                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520                         { docurl } \l_tmpb_tl
521                 }
522                 {source-base} {
523                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524                         { ns } \l_tmpb_tl
525                 }

```

```

526     {ns} {
527         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
528         { ns } \l_tmpb_tl
529     }
530     {dependencies} {
531         \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
532         { deps } \l_tmpb_tl
533     }
534     }{}{}
535 }{}
536 }
537 \ior_close:N \c__stex_mathhub_manifest_ior
538 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

539 \cs_new_protected:Nn \stex_set_current_repository:n {
540     \stex_require_repository:n { #1 }
541     \prop_set_eq:Nc \l_stex_current_repository_prop {
542         c_stex_mathhub_#1_manifest_prop
543     }
544 }

```

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

`\stex_require_repository:n`

```

545 \cs_new_protected:Nn \stex_require_repository:n {
546     \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
547         \stex_debug:nn{mathhub}{Opening~archive:~#1}
548         \_stex_mathhub_do_manifest:n { #1 }
549     }
550 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

551 %\prop_new:N \l_stex_current_repository_prop
552
553 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
554 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
555     \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
556 } {
557     \_stex_mathhub_parse_manifest:n { main }
558     \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
559     \l_tmpa_str
560     \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
561     \c_stex_mathhub_main_manifest_prop
562     \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
563     \stex_debug:nn{mathhub}{Current~repository:~
564         \prop_item:Nn \l_stex_current_repository_prop {id}
565     }
566 }

```

(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.

```

567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \prop_if_exist:NTF \l_stex_current_repository_prop {
572       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
573       \exp_args:Ne \l_tmpa_cs{
574         \prop_item:Nn \l_stex_current_repository_prop { id }
575       }
576     }{
577       \l_tmpa_cs{}
578     }
579   }{
580     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
581     \stex_require_repository:n \l_tmpa_str
582     \str_set:Nx \l_tmpa_str { #1 }
583     \exp_args:Nne \use:nn {
584       \stex_set_current_repository:n \l_tmpa_str
585       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
586     }{
587       \stex_debug:nn{mathhub}{switching~back~to:~
588       \prop_if_exist:NTF \l_stex_current_repository_prop {
589         \prop_item:Nn \l_stex_current_repository_prop { id }::~
590       \meaning\l_stex_current_repository_prop
591       }{
592         no~repository
593       }
594     }
595     \prop_if_exist:NTF \l_stex_current_repository_prop {
596       \stex_set_current_repository:n {
597         \prop_item:Nn \l_stex_current_repository_prop { id }
598       }
599     }{
600       \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
601     }
602   }
603 }
604 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

605 \def \mhpath #1 #2 {
606   \exp_args:Ne \tl_if_empty:nTF{#1}{
607     \c_stex_mathhub_str /
608     \prop_item:Nn \l_stex_current_repository_prop { id }
609     / source / #2
610   }{
611     \c_stex_mathhub_str / #1 / source / #2

```

```

612 }
613 }

```

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

`\inputref`
`\mhinput`

```

614 \newif \ifinputref \inputreffalse
615
616 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617   \stex_in_repository:nn {#1} {
618     \ifinputref
619       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620     \else
621       \inputreftrue
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \inputreffalse
624   \fi
625 }
626 }
627 \NewDocumentCommand \mhinput { 0{} m}{
628   \stex_mhinput:nn{ #1 }{ #2 }
629 }
630
631 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632   \stex_in_repository:nn {#1} {
633     \bool_lazy_any:nTF {
634       {\rustex_if_p:}
635       {\latexml_if_p:}
636     } {
637       \str_clear:N \l_tmpa_str
638       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640       }
641       \stex_annotate_invisible:nnn{inputref}{
642         \l_tmpa_str / #2
643       }{}
644     }{
645       \begingroup
646         \inputreftrue
647         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
648       \endgroup
649     }
650   }
651 }
652 \NewDocumentCommand \inputref { 0{} m}{
653   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
654 }

```

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

`\addmhbibresource`

```

655 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
656   \stex_in_repository:nn {#1} {
657     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
658   }

```

```

659 }
660 \newcommand\addmhbibresource[2][]{
661   \_stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
662 }

```

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

\libinput

```

663 \cs_new_protected:Npn \libinput #1 {
664   \prop_if_exist:NF \l_stex_current_repository_prop {
665     \msg_error:nnn{stex}{error/notinarchive}\libinput
666   }
667   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
668     \msg_error:nnn{stex}{error/notinarchive}\libinput
669   }
670   \seq_clear:N \l__stex_mathhub_libinput_files_seq
671   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
672   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
673
674   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
675     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
676     \IfFileExists{ \l_tmpa_str }{
677       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
678     }{}
679     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
680     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
681   }
682
683   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
684   \IfFileExists{ \l_tmpa_str }{
685     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
686   }{}
687
688   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
689     \msg_error:nxxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
690   }{
691     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
692       \input{ ##1 }
693     }
694   }
695 }

```

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

\libusepackage

```

696 \NewDocumentCommand \libusepackage {0{} m} {
697   \prop_if_exist:NF \l_stex_current_repository_prop {
698     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
699   }
700   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
701     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
702   }
703   \seq_clear:N \l__stex_mathhub_libinput_files_seq
704   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
705   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str

```

```

706
707 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
708   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
709   \IfFileExists{ \l_tmpa_str.sty }{
710     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
711   }{}
712   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
713   \seq_put_right:No \l_tmpa_seq \l_tmpa_str
714 }
715
716 \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
717 \IfFileExists{ \l_tmpa_str.sty }{
718   \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
719 }{}
720
721 \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
722   \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
723 }{
724   \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
725     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
726       \usepackage[#1]{ #1 }
727     }
728   }{
729     \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
730   }
731 }
732 }

```

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

`\mhgraphics`
`\cmhgraphics`

```

733
734 \AddToHook{begindocument}{
735   \ltx@ifpackageloaded{graphicx}{
736     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
737     \newcommand\mhgraphics[2][]{%
738       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
739       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}}
740     \newcommand\cmhgraphics[2][]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
741   }{}

```

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

`\lstinputmhlisting`
`\clstinputmhlisting`

```

742 \ltx@ifpackageloaded{listings}{
743   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
744   \newcommand\lstinputmhlisting[2][]{%
745     \def\lst@mhrepos{}\setkeys{lst}{#1}%
746     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}}
747   \newcommand\clstinputmhlisting[2][]{\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
748 }{}
749 }
750
751 </package>

```


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

Chapter 26

STEX -References Implementation

```
752 <*package>
753
754 %%%%%%%%%%% references.dtx %%%%%%%%%%%
755
756 <@@=stex_refs>
    Warnings and error messages
757
```

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

```
758 %\iow_new:N \c__stex_refs_refs_iow
759 \AddToHook{begindocument}{
760 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
761 }
762 \AddToHook{enddocument}{
763 % \iow_close:N \c__stex_refs_refs_iow
764 }
```

`\STEXreftitle`

```
765 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
766
767 \NewDocumentCommand \STEXreftitle { m } {
768 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
769 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
770 \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:`

```
771 \cs_new_protected:Nn \stex_get_document_uri: {  
772   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
773   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
774   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
775   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
776   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
777  
778   \str_clear:N \l_tmpa_str  
779   \prop_if_exist:NT \l_stex_current_repository_prop {  
780     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
781       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
782     }  
783   }  
784  
785   \str_if_empty:NTF \l_tmpa_str {  
786     \str_set:Nx \l_stex_current_docns_str {  
787       file:/\stex_path_to_string:N \l_tmpa_seq  
788     }  
789   }{  
790     \bool_set_true:N \l_tmpa_bool  
791     \bool_while_do:Nn \l_tmpa_bool {  
792       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
793       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
794         {source} { \bool_set_false:N \l_tmpa_bool }  
795       }{}{  
796         \seq_if_empty:NT \l_tmpa_seq {  
797           \bool_set_false:N \l_tmpa_bool  
798         }  
799       }  
800     }  
801  
802     \seq_if_empty:NTF \l_tmpa_seq {  
803       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
804     }{  
805       \str_set:Nx \l_stex_current_docns_str {  
806         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
807       }  
808     }  
809   }  
810 }
```

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

`\l_stex_current_docurl_str`

```
811 \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:`

```
812 \cs_new_protected:Nn \stex_get_document_url: {  
813   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
814   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
815   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

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

```

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

26.2 Setting Reference Targets

```

854 \str_const:Nn \c__stex_refs_url_str{URL}
855 \str_const:Nn \c__stex_refs_ref_str{REF}
856 \str_new:N \l__stex_refs_curr_label_str
857 % @currentlabel -> number
858 % @currentlabelname -> title
859 % @currentHref -> name.number <- id of some kind
860 % \theH# -> \arabic{section}
861 % \the# -> number
862 % \hyper@makecurrent{#}
863 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

864 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
865   \stex_get_document_uri:
866   \str_clear:N \l__stex_refs_curr_label_str
867   \str_set:Nx \l_tmpa_str { #1 }
868   \str_if_empty:NT \l_tmpa_str {
869     \int_incr:N \l__stex_refs_unnamed_counter_int
870     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
871   }
872   \str_set:Nx \l__stex_refs_curr_label_str {
873     \l_stex_current_docns_str?\l_tmpa_str
874   }
875   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
876     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
877   }
878   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
879     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
880   }
881   \stex_if_smsmode:TF {
882     \stex_get_document_url:
883     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
884     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
885   }{
886     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
887     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
888     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
889     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
890   }
891 }

```

(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.

```

892 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
893   \str_set:Nn \l_tmpa_str {#1?#2}
894   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
895   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
896     \seq_new:c {g__stex_refs_labels_#2_seq}
897   }
898   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
899     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
900   }
901 }

```

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

```

902 \AtEndDocument{
903   \def\stexauxadddocref#1 #2 {}{}
904 }

```

`\stex_ref_new_sym_target:n`

```

905 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
906   \stex_if_smsmode:TF {
907     \str_if_exist:cF{sref_sym_#1_type}{
908       \stex_get_document_url:
909       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

910     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
911   }
912 }{
913   \str_if_empty:NF \l__stex_refs_curr_label_str {
914     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
915     \immediate\write\@auxout{
916       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
917         \l__stex_refs_curr_label_str
918       }
919     }
920   }
921 }
922 }

```

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

26.3 Using References

```

923 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

924
925 \keys_define:nn { stex / sref } {
926   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
927   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
928   pre           .tl_set:N = \l__stex_refs_pre_tl ,
929   post          .tl_set:N = \l__stex_refs_post_tl ,
930 }
931 \cs_new_protected:Nn \__stex_refs_args:n {
932   \tl_clear:N \l__stex_refs_linktext_tl
933   \tl_clear:N \l__stex_refs_fallback_tl
934   \tl_clear:N \l__stex_refs_pre_tl
935   \tl_clear:N \l__stex_refs_post_tl
936   \str_clear:N \l__stex_refs_repo_str
937   \keys_set:nn { stex / sref } { #1 }
938 }

```

The actual macro:

```

939 \NewDocumentCommand \sref { 0{} m}{
940   \__stex_refs_args:n { #1 }
941   \str_if_empty:NTF \l__stex_refs_indocument_str {
942     \str_set:Nx \l_tmpa_str { #2 }
943     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
944     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
945       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
946         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
947           \str_clear:N \l_tmpa_str
948         }
949       }{
950         \str_clear:N \l_tmpa_str
951       }
952     }{
953       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
954       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

955 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
956 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
957   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
958   \str_clear:N \l_tmpa_str
959   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
960     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
961       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
962     }{
963       \seq_map_break:n {
964         \str_set:Nn \l_tmpa_str { ##1 }
965       }
966     }
967   }
968 }{
969   \str_clear:N \l_tmpa_str
970 }
971 }
972 \str_if_empty:NTF \l_tmpa_str {
973   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
974 }{
975   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
976     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
977       \cs_if_exist:cTF{autoref}{
978         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
979       }{
980         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
981       }
982     }{
983       \ltx@ifpackageloaded{hyperref}{
984         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
985       }{
986         \l__stex_refs_linktext_tl
987       }
988     }
989   }{
990     \ltx@ifpackageloaded{hyperref}{
991       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
992     }{
993       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
994     }
995   }
996 }
997 }{
998   % TODO
999 }
1000 }

```

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

\srefsym

```

1001 \NewDocumentCommand \srefsym { 0{} m}{
1002   \stex_get_symbol:n { #2 }
1003   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1004 }

```

```

1005
1006 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1007   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1008     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1009   }{
1010     \__stex_refs_args:n { #1 }
1011     \str_if_empty:NTF \l__stex_refs_indocument_str {
1012       \tl_if_exist:cTF{sref_sym_#2 _type}{
1013         % doc uri in \l_tmpb_str
1014         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1015         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1016           % reference
1017           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1018             \cs_if_exist:cTF{autoref}{
1019               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1020             }{
1021               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1022             }
1023           }{
1024             \ltx@ifpackageloaded{hyperref}{
1025               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1026             }{
1027               \l__stex_refs_linktext_tl
1028             }
1029           }
1030         }{
1031           % URL
1032           \ltx@ifpackageloaded{hyperref}{
1033             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1034           }{
1035             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1036           }
1037         }
1038       }{
1039         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1040       }
1041     }{
1042       % TODO
1043     }
1044   }
1045 }

```

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

\srefsymuri

```

1046 \cs_new_protected:Npn \srefsymuri #1 #2 {
1047   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1048 }

```

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

```

1049 </package>

```


Chapter 27

STEX -Modules Implementation

```
1050 <*package>
1051
1052 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1053
1054 <@@=stex_modules>
1055
1056   Warnings and error messages
1057   \msg_new:nnn{stex}{error/unknownmodule}{
1058     No~module~#1~found
1059   }
1060   \msg_new:nnn{stex}{error/syntax}{
1061     Syntax~error:~#1
1062   }
1063   \msg_new:nnn{stex}{error/siglanguage}{
1064     Module~#1~declares~signature~#2,~but~does~not~
1065     declare~its~language
1066   }
1067   \msg_new:nnn{stex}{warning/deprecated}{
1068     #1~is~deprecated;~please~use~#2~instead!
1069   }
1070   \msg_new:nnn{stex}{error/conflictingmodules}{
1071     Conflicting~imports~for~module~#1
1072   }
1073
1074 \l_stex_current_module_str The current module:
1075 \str_new:N \l_stex_current_module_str
1076
1077 (End definition for \l_stex_current_module_str. This variable is documented on page 54.)
1078
1079 \l_stex_all_modules_seq Stores all available modules
1080 \seq_new:N \l_stex_all_modules_seq
1081
1082 (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
1074 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1075   \str_if_empty:NTF \l_stex_current_module_str
1076   \prg_return_false: \prg_return_true:
1077 }

```

(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
1078 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1079   \prop_if_exist:cTF { c_stex_module_#1_prop }
1080   \prg_return_true: \prg_return_false:
1081 }

```

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

`\stex_add_to_current_module:n` Only allowed within modules:

```

\STEXexport
1082 \cs_new_protected:Nn \stex_add_to_current_module:n {
1083   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1084 }
1085 \cs_new_protected:Npn \STEXexport {
1086   \begingroup
1087   \newlinechar=-1\relax
1088   \endlinechar=-1\relax
1089   %\catcode'\ = 9\relax
1090   \expandafter\endgroup\__stex_modules_export:n
1091 }
1092 \cs_new_protected:Nn \__stex_modules_export:n {
1093   \ignorespaces #1
1094   \stex_add_to_current_module:n { \ignorespaces #1 }
1095   \stex_smsmode_do:
1096 }
1097 \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
1098 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1099   \str_set:Nx \l_tmpa_str { #1 }
1100   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1101 }

```

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

```

\stex_add_import_to_current_module:n
1102 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \exp_args:Nno
1105   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1106     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1107   }
1108 }

```

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

`\stex_collect_imports:n`

```

1109 \cs_new_protected:Nn \stex_collect_imports:n {
1110   \seq_clear:N \l_stex_collect_imports_seq
1111   \__stex_modules_collect_imports:n {#1}
1112 }
1113 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1114   \seq_map_inline:cn {c_stex_module_#1_imports} {
1115     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1116       \__stex_modules_collect_imports:n { ##1 }
1117     }
1118   }
1119   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1120     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1121   }
1122 }

```

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

`\stex_do_up_to_module:n`

```

1123 \int_new:N \l__stex_modules_group_depth_int
1124 \tl_new:N \l__stex_modules_aftergroup_tl
1125 \cs_new_protected:Nn \stex_do_up_to_module:n {
1126   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1127     #1
1128   }{
1129     #1
1130     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1131       \aftergroup\__stex_modules_aftergroup_do:
1132     }
1133   }
1134   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1135     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1136       \l__stex_modules_aftergroup_tl
1137       \tl_clear:N \l__stex_modules_aftergroup_tl
1138     }{
1139       \l__stex_modules_aftergroup_tl
1140       \aftergroup\__stex_modules_aftergroup_do:
1141     }
1142   }

```

(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).

1143

(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.

```

1144 \str_new:N \l_stex_modules_ns_str
1145 \str_new:N \l_stex_modules_subpath_str

```

```

1146 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1147   \str_set:Nx \l_tmpa_str { #1 }
1148   \seq_set_eq:NN \l_tmpa_seq #2
1149   % split off file extension
1150   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1151   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1152   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1153   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1154
1155   \bool_set_true:N \l_tmpa_bool
1156   \bool_while_do:Nn \l_tmpa_bool {
1157     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1158     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1159       {source} { \bool_set_false:N \l_tmpa_bool }
1160     }{}{
1161       \seq_if_empty:NT \l_tmpa_seq {
1162         \bool_set_false:N \l_tmpa_bool
1163       }
1164     }
1165   }
1166
1167   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1168   \str_if_empty:NTF \l_stex_modules_subpath_str {
1169     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1170   }{
1171     \str_set:Nx \l_stex_modules_ns_str {
1172       \l_tmpa_str/\l_stex_modules_subpath_str
1173     }
1174   }
1175 }
1176
1177 \cs_new_protected:Nn \stex_modules_current_namespace: {
1178   \str_clear:N \l_stex_modules_subpath_str
1179   \prop_if_exist:NTF \l_stex_current_repository_prop {
1180     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1181     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1182   }{
1183     % split off file extension
1184     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1185     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1186     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1187     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1188     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1189     \str_set:Nx \l_stex_modules_ns_str {
1190       file:\stex_path_to_string:N \l_tmpa_seq
1191     }
1192   }
1193 }

```

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

27.1 The smodule environment

smodule arguments:

```

1194 \keys_define:nn { stex / module } {
1195   title      .tl_set:N      = \smodulename ,
1196   type       .str_set_x:N   = \smodulename ,
1197   id         .str_set_x:N   = \smoduleid ,
1198   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1199   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1200   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1201   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1202   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1203   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1204   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1205   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1206 }
1207
1208 \cs_new_protected:Nn \__stex_modules_args:n {
1209   \str_clear:N \smodulename
1210   \str_clear:N \smodulename
1211   \str_clear:N \smoduleid
1212   \str_clear:N \l_stex_module_ns_str
1213   \str_clear:N \l_stex_module_deprecate_str
1214   \str_clear:N \l_stex_module_lang_str
1215   \str_clear:N \l_stex_module_sig_str
1216   \str_clear:N \l_stex_module_creators_str
1217   \str_clear:N \l_stex_module_contributors_str
1218   \str_clear:N \l_stex_module_meta_str
1219   \str_clear:N \l_stex_module_srccite_str
1220   \keys_set:nn { stex / module } { #1 }
1221 }
1222
1223 % module parameters here? In the body?
1224
```

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

```

1225 \cs_new_protected:Nn \stex_module_setup:nn {
1226   \tl_clear:N \l__stex_modules_aftergroup_tl
1227   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1228   \str_set:Nx \l_stex_module_name_str { #2 }
1229   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1230 \stex_if_in_module:TF {
1231   % Nested module
1232   \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1233   { ns } \l_stex_module_ns_str
1234   \str_set:Nx \l_stex_module_name_str {
1235     \prop_item:cn {c_stex_module\l_stex_current_module_str _prop}
1236     { name } / \l_stex_module_name_str
1237   }
1238 }{
1239   % not nested:

```

```

1240 \str_if_empty:NT \l_stex_module_ns_str {
1241   \stex_modules_current_namespace:
1242   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1243   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1244     / {\l_stex_module_ns_str}
1245   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1246   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1247     \str_set:Nx \l_stex_module_ns_str {
1248       \stex_path_to_string:N \l_tmpa_seq
1249     }
1250   }
1251 }
1252 }

```

Next, we determine the language of the module:

```

1253 \str_if_empty:NT \l_stex_module_lang_str {
1254   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1255   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1256   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1257   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1258   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1259     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1260       inferred~from~file~name}
1261     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1262   }
1263 }
1264
1265 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1266   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1267   \l_tmpa_str {
1268     \ltx@ifpackageloaded{babel}{
1269       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1270     }{}
1271   } {
1272     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1273   }
1274 }}

```

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

```

1275 \str_if_empty:NTF \l_stex_module_sig_str {
1276   \exp_args:Nnx \prop_gset_from_keyval:cn {
1277     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1278   } {
1279     name      = \l_stex_module_name_str ,
1280     ns        = \l_stex_module_ns_str ,
1281     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1282     lang      = \l_stex_module_lang_str ,
1283     sig       = \l_stex_module_sig_str ,
1284     deprecate = \l_stex_module_deprecate_str ,
1285     meta      = \l_stex_module_meta_str
1286   }
1287   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1288   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}

```

```

1289 \tl_clear:c {c_stex_module\_l_stex_module_ns_str?\l_stex_module_name_str _code}
1290 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1291 \str_if_empty:NT \l_stex_module_meta_str {
1292   \str_set:Nx \l_stex_module_meta_str {
1293     \c_stex_metatheory_ns_str ? Metatheory
1294   }
1295 }
1296 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1297   \bool_set_true:N \l_stex_in_meta_bool
1298   \exp_args:Nx \stex_add_to_current_module:n {
1299     \bool_set_true:N \l_stex_in_meta_bool
1300     \stex_activate_module:n {\l_stex_module_meta_str}
1301     \bool_set_false:N \l_stex_in_meta_bool
1302   }
1303   \stex_activate_module:n {\l_stex_module_meta_str}
1304   \bool_set_false:N \l_stex_in_meta_bool
1305 }
1306 }{
1307   \str_if_empty:NT \l_stex_module_lang_str {
1308     \msg_error:nnxx{stex}{error/siglanguage}{
1309       \l_stex_module_ns_str?\l_stex_module_name_str
1310     }{\l_stex_module_sig_str}
1311   }
1312
1313   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1314   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1315   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1316   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1317   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1318   \str_set:Nx \l_tmpa_str {
1319     \stex_path_to_string:N \l_tmpa_seq /
1320     \l_tmpa_str . \l_stex_module_sig_str .tex
1321   }
1322   \IfFileExists \l_tmpa_str {
1323     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1324       \str_clear:N \l_stex_current_module_str
1325       \seq_clear:N \l_stex_all_modules_seq
1326       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1327     }
1328   }{
1329     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1330   }
1331   \stex_if_smsmode:F {
1332     \stex_activate_module:n {
1333       \l_stex_module_ns_str ? \l_stex_module_name_str
1334     }
1335   }
1336   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1337 }
1338 \str_if_empty:NF \l_stex_module_deprecate_str {
1339   \msg_warning:nnxx{stex}{warning/deprecated}{
1340     Module~\l_stex_current_module_str

```

```

1341     }{
1342       \l_stex_module_deprecate_str
1343     }
1344   }
1345   \seq_put_right:Nx \l_stex_all_modules_seq {
1346     \l_stex_module_ns_str ? \l_stex_module_name_str
1347   }
1348 }

```

(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}`

```

1349 \cs_new_protected:Nn \_stex_modules_begin_module: {
1350   \stex_reactivate_macro:N \STEXexport
1351   \stex_reactivate_macro:N \importmodule
1352   \stex_reactivate_macro:N \symdecl
1353   \stex_reactivate_macro:N \notation
1354   \stex_reactivate_macro:N \symdef
1355
1356   \stex_debug:nn{modules}{
1357     New~module:\
1358     Namespace:~\l_stex_module_ns_str\
1359     Name:~\l_stex_module_name_str\
1360     Language:~\l_stex_module_lang_str\
1361     Signature:~\l_stex_module_sig_str\
1362     Metatheory:~\l_stex_module_meta_str\
1363     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1364   }
1365
1366   \stex_if_smsmode:F{
1367     \begin{stex_annotate_env} {theory} {
1368       \l_stex_module_ns_str ? \l_stex_module_name_str
1369     }
1370
1371     \stex_annotate_invisible:nnn{header}{} {
1372       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1373       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1374       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1375         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1376       }
1377       \str_if_empty:NF \smodulotype {
1378         \stex_annotate:nnn{type}{\smodulotype}{}
1379       }
1380     }
1381   }
1382   % TODO: Inherit metatheory for nested modules?
1383 }
1384 \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{smodule}`


```

1385 \cs_new_protected:Nn \__stex_modules_end_module: {
1386   \stex_debug:nn{modules}{Closing-module~\prop_item:cn {c_stex_module_\l_stex_current_module}}
1387 }

```

(End definition for __stex_modules_end_module:.)

The core environment

```

1388 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1389 \NewDocumentEnvironment { smodule } { 0{} m } {
1390   \stex_module_setup:nn{#1}{#2}
1391   \par
1392   \stex_if_smsmode:F{
1393     \tl_clear:N \l_tmpa_tl
1394     \clist_map_inline:Nn \smodulotype {
1395       \tl_if_exist:cT {\__stex_modules_smodule_##1_start:}{
1396         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_start:}}
1397       }
1398     }
1399     \tl_if_empty:NTF \l_tmpa_tl {
1400       \__stex_modules_smodule_start:
1401     }{
1402       \l_tmpa_tl
1403     }
1404   }
1405   \__stex_modules_begin_module:
1406   \str_if_empty:NF \smoduleid {
1407     \stex_ref_new_doc_target:n \smoduleid
1408   }
1409   \stex_smsmode_do:
1410 } {
1411   \__stex_modules_end_module:
1412   \stex_if_smsmode:F {
1413     \end{stex_annotate_env}
1414     \clist_set:No \l_tmpa_clist \smodulotype
1415     \tl_clear:N \l_tmpa_tl
1416     \clist_map_inline:Nn \l_tmpa_clist {
1417       \tl_if_exist:cT {\__stex_modules_smodule_##1_end:}{
1418         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_end:}}
1419       }
1420     }
1421     \tl_if_empty:NTF \l_tmpa_tl {
1422       \__stex_modules_smodule_end:
1423     }{
1424       \l_tmpa_tl
1425     }
1426   }
1427 }

```

\stexpatchmodule

```

1428 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1429 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1430
1431 \newcommand\stexpatchmodule[3] [] {
1432   \str_set:Nx \l_tmpa_str{ #1 }
1433   \str_if_empty:NTF \l_tmpa_str {

```

```

1434     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1435     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1436   }{
1437     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1438     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1439   }
1440 }

```

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

27.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1441 \NewDocumentCommand \STEXModule { m } {
1442   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1443   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1444   \tl_set:Nn \l_tmpa_tl {
1445     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1446   }
1447   \seq_map_inline:Nn \l_stex_all_modules_seq {
1448     \str_set:Nn \l_tmpb_str { ##1 }
1449     \str_if_eq:eeT { \l_tmpa_str } {
1450       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1451     } {
1452       \seq_map_break:n {
1453         \tl_set:Nn \l_tmpa_tl {
1454           \stex_invoke_module:n { ##1 }
1455         }
1456       }
1457     }
1458   }
1459   \l_tmpa_tl
1460 }
1461
1462 \cs_new_protected:Nn \stex_invoke_module:n {
1463   \stex_debug:nn{modules}{Invoking~module~#1}
1464   \peek_charcode_remove:NTF ! {
1465     \__stex_modules_invoke_uri:nN { #1 }
1466   } {
1467     \peek_charcode_remove:NTF ? {
1468       \__stex_modules_invoke_symbol:nn { #1 }
1469     } {
1470       \msg_error:nnx{stex}{error/syntax}{
1471         ?~or~!~expected~after~
1472         \c_backslash_str STEXModule{#1}
1473       }
1474     }
1475   }
1476 }
1477
1478 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1479   \str_set:Nn #2 { #1 }
1480 }

```

```

1481
1482 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1483   \stex_invoke_symbol:n{#1?#2}
1484 }

```

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

`\stex_activate_module:n`

```

1485 \bool_new:N \l_stex_in_meta_bool
1486 \bool_set_false:N \l_stex_in_meta_bool
1487 \cs_new_protected:Nn \stex_activate_module:n {
1488   \stex_debug:nn{modules}{Activating~module~#1}
1489   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1490     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1491   }
1492   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1493     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1494     \use:c{ c_stex_module_#1_code }
1495   }
1496 }

```

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

```

1497 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1498 ⟨*package⟩
1499
1500 %%%%%%%%% inheritance.dtx %%%%%%%%%
1501
```

28.1 SMS Mode

```
1502 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1503 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1504 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1505 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1506
1507 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1508   \makeatletter
1509   \makeatother
1510   \ExplSyntaxOn
1511   \ExplSyntaxOff
1512   \rustexBREAK
1513 }
1514
1515 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1516   \symdef
1517   \importmodule
1518   \notation
1519   \symdecl
1520   \STEXexport
1521   \inlineass
1522   \inlinedef
1523   \inlineex
1524   \endinput
1525   \setnotation
```

```

1526 \copynotation
1527 }
1528
1529 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1530   \tl_to_str:n {
1531     smodule,
1532     copymodule,
1533     interpretmodule,
1534     sdefinition,
1535     sexample,
1536     sassertion,
1537     sparagraph
1538   }
1539 }

```

(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`

```

1540 \bool_new:N \g__stex_smsmode_bool
1541 \bool_set_false:N \g__stex_smsmode_bool
1542 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1543   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1544 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1545 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1546   \vbox_set:Nn \l_tmpa_box {
1547     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1548     \bool_gset_true:N \g__stex_smsmode_bool
1549     #2
1550     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1551   }
1552   \box_clear:N \l_tmpa_box
1553 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1554 \quark_new:N \q__stex_smsmode_break
1555
1556 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1557   \stex_filestack_push:n{#1}
1558   \_stex_smsmode_in_smsmode:nn{#1} {
1559     #2
1560     \everyeof{\q__stex_smsmode_break\noexpand}
1561     \expandafter\expandafter\expandafter
1562     \stex_smsmode_do:
1563     \csname @ @ input\endcsname "#1"\relax
1564   }
1565   \stex_filestack_pop:
1566 }

```

(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:

```

1567 \cs_new_protected:Npn \stex_smsmode_do: {
1568   \stex_if_smsmode:T {
1569     \__stex_smsmode_do:w
1570   }
1571 }
1572 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1573   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1574     \expandafter\if\expandafter\relax\noexpand#1
1575     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1576   }else\expandafter\__stex_smsmode_do:w\fi
1577 }{
1578   \__stex_smsmode_do:w % #1
1579 }
1580 }
1581 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1582   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1583     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1584       #1\__stex_smsmode_do:w
1585     }{
1586       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1587         #1
1588       }{
1589         \cs_if_eq:NNTF \begin #1 {
1590           \__stex_smsmode_check_begin:n
1591         }{
1592           \cs_if_eq:NNTF \end #1 {
1593             \__stex_smsmode_check_end:n
1594           }{
1595             \__stex_smsmode_do:w
1596           }
1597         }
1598       }
1599     }
1600   }
1601 }
1602
1603 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1604   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1605     \begin{#1}
1606   }{
1607     \__stex_smsmode_do:w
1608   }
1609 }
1610 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1611   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1612     \end{#1}\__stex_smsmode_do:w
1613   }{
1614     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1615   }
1616 }

```

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

28.2 Inheritance

```

1617 <@@=stex_importmodule>

\stex_import_module_uri:nn

1618 \cs_new_protected:Nn \stex_import_module_uri:nn {
1619   \str_set:Nx \l_stex_import_archive_str { #1 }
1620   \str_set:Nn \l_stex_import_path_str { #2 }
1621
1622   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1623   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1624   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1625
1626   \stex_modules_current_namespace:
1627   \bool_lazy_all:nTF {
1628     {\str_if_empty_p:N \l_stex_import_archive_str}
1629     {\str_if_empty_p:N \l_stex_import_path_str}
1630     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1631   }{
1632     \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1633     \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1634   }{
1635     \str_if_empty:NT \l_stex_import_archive_str {
1636       \prop_if_exist:NT \l_stex_current_repository_prop {
1637         \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1638       }
1639     }
1640     \str_if_empty:NTF \l_stex_import_archive_str {
1641       \str_if_empty:NF \l_stex_import_path_str {
1642         \str_set:Nx \l_stex_import_ns_str {
1643           \l_stex_module_ns_str / \l_stex_import_path_str
1644         }
1645       }
1646     }{
1647       \stex_require_repository:n \l_stex_import_archive_str
1648       \prop_get:cnN { c_stex_mathhub \l_stex_import_archive_str _manifest_prop } { ns }
1649       \l_stex_import_ns_str
1650       \str_if_empty:NF \l_stex_import_path_str {
1651         \str_set:Nx \l_stex_import_ns_str {
1652           \l_stex_import_ns_str / \l_stex_import_path_str
1653         }
1654       }
1655     }
1656   }
1657 }
```

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

<code>\l_stex_import_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l_stex_import_archive_str</code>	<code>\str_new:N \l_stex_import_name_str</code>
<code>\l_stex_import_path_str</code>	<code>\str_new:N \l_stex_import_archive_str</code>
<code>\l_stex_import_ns_str</code>	<code>\str_new:N \l_stex_import_path_str</code>

```
1661 \str_new:N \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:nnnn {{ns}} {{archive-ID}} {{path}} {{name}}

1662 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1663   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1664
1665     % archive
1666     \str_set:Nx \l_tmpa_str { #2 }
1667     \str_if_empty:NTF \l_tmpa_str {
1668       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1669     } {
1670       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1671       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1672       \seq_put_right:Nn \l_tmpa_seq { source }
1673     }
1674
1675     % path
1676     \str_set:Nx \l_tmpb_str { #3 }
1677     \str_if_empty:NTF \l_tmpb_str {
1678       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1679
1680       \ltx@ifpackageloaded{babel} {
1681         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1682           { \language } \l_tmpb_str {
1683           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1684         }
1685       } {
1686         \str_clear:N \l_tmpb_str
1687       }
1688
1689       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1690       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1691         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1692       }{
1693         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1694         \IfFileExists{ \l_tmpa_str.tex }{
1695           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1696         }{
1697           % try english as default
1698           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1699           \IfFileExists{ \l_tmpa_str.en.tex }{
1700             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1701           }{
1702             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1703           }
1704         }
1705       }
1706
1707     } {
1708       \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1709       \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1710
```



```

1711 \ltx@ifpackageloaded{babel} {
1712   \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1713     { \language } \l_tmpb_str {
1714       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1715     }
1716   } {
1717     \str_clear:N \l_tmpb_str
1718   }
1719
1720   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1721
1722   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1723   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1724     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1725   }{
1726     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1727     \IfFileExists{ \l_tmpa_str/#4.tex }{
1728       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1729     }{
1730       % try english as default
1731       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1732       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1733         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1734       }{
1735         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1736         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1737           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1738         }{
1739           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1740           \IfFileExists{ \l_tmpa_str.tex }{
1741             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1742           }{
1743             % try english as default
1744             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1745             \IfFileExists{ \l_tmpa_str.en.tex }{
1746               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1747             }{
1748               \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1749             }
1750           }
1751         }
1752       }
1753     }
1754   }
1755 }
1756
1757 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1758   \seq_clear:N \l_stex_all_modules_seq
1759   \str_clear:N \l_stex_current_module_str
1760   \str_set:Nx \l_tmpb_str { #2 }
1761   \str_if_empty:NF \l_tmpb_str {
1762     \stex_set_current_repository:n { #2 }
1763   }
1764   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}

```

```

1765     }
1766
1767     \stex_if_module_exists:nF { #1 ? #4 } {
1768       \msg_error:nnx{stex}{error/unknownmodule}{
1769         #1?#4~(in~file~\g__stex_importmodule_file_str)
1770       }
1771     }
1772   }
1773   \stex_activate_module:n { #1 ? #4 }
1774 }

```

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

`\importmodule`

```

1775 \NewDocumentCommand \importmodule { 0{ } m } {
1776   \stex_import_module_uri:nn { #1 } { #2 }
1777   \stex_debug:nn{modules}{Importing~module:~
1778     \l_stex_import_ns_str ? \l_stex_import_name_str
1779   }
1780   \stex_if_smsmode:F {
1781     \stex_import_require_module:nnnn
1782     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1783     { \l_stex_import_path_str } { \l_stex_import_name_str }
1784     \stex_annotate_invisible:nnn
1785     {import} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1786   }
1787   \exp_args:Nx \stex_add_to_current_module:n {
1788     \stex_import_require_module:nnnn
1789     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1790     { \l_stex_import_path_str } { \l_stex_import_name_str }
1791   }
1792   \exp_args:Nx \stex_add_import_to_current_module:n {
1793     \l_stex_import_ns_str ? \l_stex_import_name_str
1794   }
1795   \stex_smsmode_do:
1796   \ignorespacesandpars
1797 }
1798 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

`\usemodule`

```

1799 \NewDocumentCommand \usemodule { 0{ } m } {
1800   \stex_if_smsmode:F {
1801     \stex_import_module_uri:nn { #1 } { #2 }
1802     \stex_import_require_module:nnnn
1803     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1804     { \l_stex_import_path_str } { \l_stex_import_name_str }
1805     \stex_annotate_invisible:nnn
1806     {usemodule} { \l_stex_import_ns_str ? \l_stex_import_name_str } {}
1807   }
1808   \stex_smsmode_do:
1809   \ignorespacesandpars
1810 }

```

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

1811 `\endpackage`

Chapter 29

STEX -Symbols Implementation

```
1812 <*package>
1813
1814 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1815
    Warnings and error messages
1816 \msg_new:nnn{stex}{error/wrongargs}{
1817   args~value~in~symbol~declaration~for~#1~
1818   needs~to~be~i,~a,~b~or~B,~but~#2~given
1819 }
1820 \msg_new:nnn{stex}{error/unknownsymbol}{
1821   No~symbol~#1~found!
1822 }
1823 \msg_new:nnn{stex}{error/seqlength}{
1824   Expected~#1~arguments;~got~#2!
1825 }
```

29.1 Symbol Declarations

```
1826 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1827 \cs_new_protected:Nn \stex_all_symbols:n {
1828   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1829   \seq_map_inline:Nn \l_stex_all_modules_seq {
1830     \seq_map_inline:cn{c_stex_module_##1_constants}{
1831       \__stex_symdecl_all_symbols_cs{##1?####1}
1832     }
1833   }
1834 }

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

\STEXsymbol
1835 \NewDocumentCommand \STEXsymbol { m } {
1836   \stex_get_symbol:n { #1 }
```

```

1837 \exp_args:No
1838 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1839 }

```

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

`symdecl` arguments:

```

1840 \keys_define:nn { stex / symdecl } {
1841   name      .str_set:N = \l_stex_symdecl_name_str ,
1842   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1843   args      .str_set:N = \l_stex_symdecl_args_str ,
1844   type      .tl_set:N  = \l_stex_symdecl_type_tl ,
1845   deprecate .str_set:N = \l_stex_symdecl_deprecate_str ,
1846   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1847   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1848   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1849   def       .tl_set:N  = \l_stex_symdecl_definiens_tl ,
1850   assoc     .choices:nn =
1851     {bin,binl,binr,pre,conj,pwconj}
1852     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1853 }
1854
1855 \bool_new:N \l_stex_symdecl_make_macro_bool
1856
1857 \cs_new_protected:Nn \__stex_symdecl_args:n {
1858   \str_clear:N \l_stex_symdecl_name_str
1859   \str_clear:N \l_stex_symdecl_args_str
1860   \str_clear:N \l_stex_symdecl_deprecate_str
1861   \str_clear:N \l_stex_symdecl_astype_str
1862   \bool_set_false:N \l_stex_symdecl_local_bool
1863   \tl_clear:N \l_stex_symdecl_type_tl
1864   \tl_clear:N \l_stex_symdecl_definiens_tl
1865
1866   \keys_set:nn { stex / symdecl } { #1 }
1867 }

```

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

```

1868
1869 \NewDocumentCommand \symdecl { s m O{} } {
1870   \__stex_symdecl_args:n { #3 }
1871   \IfBooleanTF #1 {
1872     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1873   } {
1874     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1875   }
1876   \stex_symdecl_do:n { #2 }
1877   \stex_smsmode_do:
1878 }
1879
1880 \cs_new_protected:Nn \stex_symdecl_do:nn {
1881   \__stex_symdecl_args:n{#1}
1882   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1883   \stex_symdecl_do:n{#2}
1884 }

```

```

1885
1886 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1887 \cs_new_protected:Nn \stex_symdecl_do:n {
1888   \stex_if_in_module:F {
1889     % TODO throw error? some default namespace?
1890   }
1891
1892   \str_if_empty:NT \l_stex_symdecl_name_str {
1893     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1894   }
1895
1896   \prop_if_exist:cT { l_stex_symdecl_
1897     \l_stex_current_module_str ?
1898     \l_stex_symdecl_name_str
1899   }_prop
1900   {
1901     % TODO throw error (beware of circular dependencies)
1902   }
1903
1904   \prop_clear:N \l_tmpa_prop
1905   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1906   \seq_clear:N \l_tmpa_seq
1907   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1908   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1909
1910   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1911     \str_if_empty:NF \l_stex_module_deprecate_str {
1912       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1913     }
1914   }
1915   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1916
1917   \exp_args:No \stex_add_constant_to_current_module:n {
1918     \l_stex_symdecl_name_str
1919   }
1920
1921   % arity/args
1922   \int_zero:N \l_tmpb_int
1923
1924   \bool_set_true:N \l_tmpa_bool
1925   \str_map_inline:Nn \l_stex_symdecl_args_str {
1926     \token_case_meaning:NnF ##1 {
1927       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1928       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1929       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1930       {\tl_to_str:n a} {
1931         \bool_set_false:N \l_tmpa_bool
1932         \int_incr:N \l_tmpb_int
1933       }
1934       {\tl_to_str:n B} {

```

```

1935     \bool_set_false:N \l_tmpa_bool
1936     \int_incr:N \l_tmpb_int
1937   }
1938 }{
1939   \msg_error:nnxx{stex}{error/wrongargs}{
1940     \l_stex_current_module_str ?
1941     \l_stex_symdecl_name_str
1942   }{##1}
1943 }
1944 }
1945 \bool_if:NTF \l_tmpa_bool {
1946   % possibly numeric
1947   \str_if_empty:NTF \l_stex_symdecl_args_str {
1948     \prop_put:Nnn \l_tmpa_prop { args } {}
1949     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1950   }{
1951     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1952     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1953     \str_clear:N \l_tmpa_str
1954     \int_step_inline:nn \l_tmpa_int {
1955       \str_put_right:Nn \l_tmpa_str i
1956     }
1957     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1958   }
1959 } {
1960   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1961   \prop_put:Nnx \l_tmpa_prop { arity }
1962     { \str_count:N \l_stex_symdecl_args_str }
1963 }
1964 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
1965
1966 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
1967   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
1968 }{
1969   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
1970 }
1971
1972 % semantic macro
1973
1974 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1975   \exp_args:Nx \stex_do_up_to_module:n {
1976     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1977       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1978     }}
1979   }
1980
1981   \bool_if:NF \l_stex_symdecl_local_bool {
1982     \exp_args:Nx \stex_add_to_current_module:n {
1983       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1984         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1985       } }
1986     }
1987   }
1988 }

```

```

1989
1990 \stex_debug:nn{symbols}{New~symbol:~
1991   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1992   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1993   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
1994   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
1995 }
1996
1997 % circular dependencies require this:
1998
1999 \prop_if_exist:cF {
2000   \l_stex_symdecl_
2001   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2002   _prop
2003 } {
2004   \exp_args:Nx \stex_do_up_to_module:n {
2005     \prop_set_from_keyval:cn {
2006       \l_stex_symdecl_
2007       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2008       _prop
2009     } {\prop_to_keyval:N \l_tmpa_prop}
2010     \seq_clear:c {
2011       \l_stex_symdecl_
2012       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2013       _notations
2014     }
2015   }
2016 }
2017
2018
2019
2020 \bool_if:NF \l_stex_symdecl_local_bool {
2021   \exp_args:Nx
2022   \stex_add_to_current_module:n {
2023     \seq_clear:c {
2024       \l_stex_symdecl_
2025       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2026       _notations
2027     }
2028     \prop_set_from_keyval:cn {
2029       \l_stex_symdecl_
2030       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2031       _prop
2032     } {
2033       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2034       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2035       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2036       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2037       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2038       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
2039     }
2040   }
2041 }
2042

```



```

2043 \stex_if_smsmode:F {
2044 % \exp_args:Nx \stex_do_up_to_module:n {
2045 % \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2046 % \l_stex_current_module_str ? \l_stex_symdecl_name_str
2047 % }
2048 % }
2049 \stex_if_do_html:T {
2050 \stex_annotate_invisible:nnn {symdecl} {
2051 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2052 } {
2053 \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{ $\l_st
2054 \stex_annotate_invisible:nnn{args}{}{
2055 \prop_item:Nn \l_tmpa_prop { args }
2056 }
2057 \stex_annotate_invisible:nnn{macroname}{#1}{}
2058 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2059 \stex_annotate_invisible:nnn{definiens}{}
2060 { $\l_stex_symdecl_definiens_tl$
2061 }
2062 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2063 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2064 }
2065 }
2066 }
2067 }
2068 }

```

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

`\stex_get_symbol:n`

```

2069 \str_new:N \l_stex_get_symbol_uri_str
2070
2071 \cs_new_protected:Nn \stex_get_symbol:n {
2072 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2073 \tl_set:Nn \l_tmpa_tl { #1 }
2074 \__stex_symdecl_get_symbol_from_cs:
2075 }{
2076 % argument is a string
2077 % is it a command name?
2078 \cs_if_exist:cTF { #1 }{
2079 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2080 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2081 \str_if_empty:NTF \l_tmpa_str {
2082 \exp_args:Nx \cs_if_eq:NNTF {
2083 \tl_head:N \l_tmpa_tl
2084 } \stex_invoke_symbol:n {
2085 \__stex_symdecl_get_symbol_from_cs:
2086 }{
2087 \__stex_symdecl_get_symbol_from_string:n { #1 }
2088 }
2089 } {
2090 \__stex_symdecl_get_symbol_from_string:n { #1 }
2091 }
2092 }{

```

```

2093     % argument is not a command name
2094     \__stex_symdecl_get_symbol_from_string:n { #1 }
2095     % \l_stex_all_symbols_seq
2096   }
2097 }
2098 \str_if_eq:eeF {
2099   \prop_item:cn {
2100     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2101   }{ deprecate }
2102 }{}{
2103   \msg_warning:nnxx{stex}{warning/deprecated}{
2104     Symbol~\l_stex_get_symbol_uri_str
2105   }{
2106     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2107   }
2108 }
2109 }
2110
2111 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2112   \tl_set:Nn \l_tmpa_tl {
2113     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2114   }
2115   \str_set:Nn \l_tmpa_str { #1 }
2116   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2117
2118   \stex_all_symbols:n {
2119     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2120       \seq_map_break:n{\seq_map_break:n{
2121         \tl_set:Nn \l_tmpa_tl {
2122           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2123         }
2124       }}
2125     }
2126   }
2127
2128   \l_tmpa_tl
2129 }
2130
2131 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2132   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2133     { \tl_tail:N \l_tmpa_tl }
2134   \tl_if_single:NTF \l_tmpa_tl {
2135     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2136       \exp_after:wN \str_set:Nn \exp_after:wN
2137         \l_stex_get_symbol_uri_str \l_tmpa_tl
2138     }{
2139       % TODO
2140       % tail is not a single group
2141     }
2142   }{
2143     % TODO
2144     % tail is not a single group
2145   }
2146 }

```

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

29.2 Notations

2147 `<@@=stex_notation>`

notation arguments:

```
2148 \keys_define:nn { stex / notation } {
2149   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2150   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2151   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2152   op        .tl_set:N   = \l__stex_notation_op_tl ,
2153   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2154   primary   .default:n  = {true} ,
2155   unknown   .code:n     = \str_set:Nx
2156               \l__stex_notation_variant_str \l_keys_key_str
2157 }
2158
2159 \cs_new_protected:Nn \_stex_notation_args:n {
2160   \str_clear:N \l__stex_notation_lang_str
2161   \str_clear:N \l__stex_notation_variant_str
2162   \str_clear:N \l__stex_notation_prec_str
2163   \tl_clear:N \l__stex_notation_op_tl
2164   \bool_set_false:N \l__stex_notation_primary_bool
2165
2166   \keys_set:nn { stex / notation } { #1 }
2167 }
```

`\notation`

```
2168 \NewDocumentCommand \notation { s m O{}} {
2169   \_stex_notation_args:n { #3 }
2170   \tl_clear:N \l_stex_symdecl_definiens_tl
2171   \stex_get_symbol:n { #2 }
2172   \tl_set:Nn \l_stex_notation_after_do_tl {
2173     \__stex_notation_final:
2174     \IfBooleanTF#1{
2175       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2176     }{}
2177     \stex_smsmode_do:\ignorespacesandpars
2178   }
2179   \stex_notation_do:nnnnn
2180     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2181     { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2182     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2183     { \l__stex_notation_prec_str }
2184   }
2185   \stex_deactivate_macro:Nn \notation {module~environments}
```

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

`\stex_notation_do:nnnnn`

```
2186 \seq_new:N \l__stex_notation_precedences_seq
2187 \tl_new:N \l__stex_notation_opprec_tl
2188 \int_new:N \l__stex_notation_currarg_int
```

```

2189 \tl_new:N \stex_symbol_after_invokation_tl
2190
2191 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2192   \let\l_stex_current_symbol_str\relax
2193   \seq_clear:N \l__stex_notation_precedences_seq
2194   \tl_clear:N \l__stex_notation_opprec_tl
2195   \str_set:Nx \l__stex_notation_args_str { #1 }
2196   \str_set:Nx \l__stex_notation_arity_str { #2 }
2197   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2198   \str_set:Nx \l__stex_notation_prec_str { #4 }
2199
2200   % precedences
2201   \str_if_empty:NTF \l__stex_notation_prec_str {
2202     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2203       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2204     }{
2205       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2206     }
2207   } {
2208     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2209       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2210       \int_step_inline:nn { \l__stex_notation_arity_str } {
2211         \exp_args:NNo
2212         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2213       }
2214     }{
2215       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2216       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2217         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2218         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2219           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2220             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2221           \seq_map_inline:Nn \l_tmpa_seq {
2222             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2223           }
2224         }
2225       }{
2226         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2227           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2228         }{
2229           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2230         }
2231       }
2232     }
2233   }
2234
2235   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2236   \int_step_inline:nn { \l__stex_notation_arity_str } {
2237     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2238       \exp_args:NNo
2239       \seq_put_right:No \l__stex_notation_precedences_seq {
2240         \l__stex_notation_opprec_tl
2241       }
2242     }
  
```

```

2243 }
2244 \tl_clear:N \l_stex_notation_dummyargs_tl
2245
2246 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2247   \exp_args:NNe
2248   \cs_set:Npn \l_stex_notation_macrocode_cs {
2249     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2250     { \l__stex_notation_suffix_str }
2251     { \l__stex_notation_opprec_tl }
2252     { \exp_not:n { #5 } }
2253   }
2254   \l_stex_notation_after_do_tl
2255 }{
2256   \str_if_in:NnTF \l__stex_notation_args_str b {
2257     \exp_args:Nne \use:nn
2258     {
2259       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2260       \cs_set:Npn \l__stex_notation_arity_str } { {
2261         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2262         { \l__stex_notation_suffix_str }
2263         { \l__stex_notation_opprec_tl }
2264         { \exp_not:n { #5 } }
2265       }}
2266   }{
2267     \str_if_in:NnTF \l__stex_notation_args_str B {
2268       \exp_args:Nne \use:nn
2269       {
2270         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2271         \cs_set:Npn \l__stex_notation_arity_str } { {
2272           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2273           { \l__stex_notation_suffix_str }
2274           { \l__stex_notation_opprec_tl }
2275           { \exp_not:n { #5 } }
2276         } }
2277     }{
2278       \exp_args:Nne \use:nn
2279       {
2280         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2281         \cs_set:Npn \l__stex_notation_arity_str } { {
2282           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2283           { \l__stex_notation_suffix_str }
2284           { \l__stex_notation_opprec_tl }
2285           { \exp_not:n { #5 } }
2286         } }
2287     }
2288   }
2289
2290   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2291   \int_zero:N \l__stex_notation_currarg_int
2292   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2293   \__stex_notation_arguments:
2294 }
2295 }

```

(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

```

2296 \cs_new_protected:Nn \_stex_notation_arguments: {
2297   \int_incr:N \l__stex_notation_currarg_int
2298   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2299     \l_stex_notation_after_do_tl
2300   }{
2301     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2302     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2303     \str_if_eq:NnTF \l_tmpa_str a {
2304       \_stex_notation_argument_assoc:n
2305     }{
2306       \str_if_eq:NnTF \l_tmpa_str B {
2307         \_stex_notation_argument_assoc:n
2308       }{
2309         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2310         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2311           { \_stex_term_math_arg:nnn
2312             { \int_use:N \l__stex_notation_currarg_int }
2313             { \l_tmpa_str }
2314             { ####\int_use:N \l__stex_notation_currarg_int }
2315           }
2316         }
2317         \_stex_notation_arguments:
2318       }
2319     }
2320   }
2321 }

```

(End definition for `_stex_notation_arguments:.`)

`_stex_notation_argument_assoc:n`

```

2322 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2323
2324   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2325     {\l__stex_notation_arity_str}{
2326       #1
2327     }
2328   \int_zero:N \l_tmpa_int
2329   \tl_clear:N \l_tmpa_tl
2330   \str_map_inline:Nn \l__stex_notation_args_str {
2331     \int_incr:N \l_tmpa_int
2332     \tl_put_right:Nx \l_tmpa_tl {
2333       \str_if_eq:nnTF {##1}{a}{ {} }{
2334         \str_if_eq:nnTF {##1}{B}{ {} }{
2335           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2336         }
2337       }
2338     }
2339   }
2340   \exp_after:wN\exp_after:wN\exp_after:wN \def
2341   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2342   \exp_after:wN\exp_after:wN\exp_after:wN ##
2343   \exp_after:wN\exp_after:wN\exp_after:wN 1
2344   \exp_after:wN\exp_after:wN\exp_after:wN ##

```

```

2345 \exp_after:wN\exp_after:wN\exp_after:wN 2
2346 \exp_after:wN\exp_after:wN\exp_after:wN {
2347   \exp_after:wN \exp_after:wN \exp_after:wN
2348   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2349     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2350   }
2351 }
2352
2353 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2354 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2355   \stex_term_math_assoc_arg:nnnn
2356   { \int_use:N \l__stex_notation_currarg_int }
2357   { \l_tmpa_str }
2358   { ####\int_use:N \l__stex_notation_currarg_int }
2359   { \l_tmpa_cs {####1} {####2} }
2360 } }
2361 \__stex_notation_arguments:
2362 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2363 \cs_new_protected:Nn \__stex_notation_final: {
2364 % \exp_args:Nne \use:nn
2365 % {
2366 % \cs_generate_from_arg_count:cNnn {
2367 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2368 %   \l__stex_notation_suffix_str
2369 %   _cs
2370 % }
2371 % \cs_set:Npn \l__stex_notation_arity_str } { {
2372 %   \exp_after:wN \exp_after:wN \exp_after:wN
2373 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2374 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2375 % } }
2376
2377 % \tl_if_empty:NF \l__stex_notation_op_tl {
2378 %   \cs_set:cpx {
2379 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2380 %     \l__stex_notation_suffix_str
2381 %     _cs
2382 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2383 % }
2384
2385 \exp_args:Nx \stex_do_up_to_module:n {
2386   \cs_generate_from_arg_count:cNnn {
2387     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2388     \l__stex_notation_suffix_str
2389     _cs
2390   } \cs_set:Npn { \l__stex_notation_arity_str } {
2391     \exp_after:wN \exp_after:wN \exp_after:wN
2392     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2393     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2394   }

```

```

2395 \tl_if_empty:NF \l__stex_notation_op_tl {
2396   \cs_set:cpn {
2397     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2398     \l__stex_notation_suffix_str
2399     _cs
2400   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2401 }
2402 }
2403
2404 \exp_args:Ne
2405 \stex_add_to_current_module:n {
2406   \cs_generate_from_arg_count:cNnn {
2407     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2408     \l__stex_notation_suffix_str
2409     _cs
2410   } \cs_set:Npn {\l__stex_notation_arity_str} {
2411     \exp_after:wN \exp_after:wN \exp_after:wN
2412     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2413     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2414   }
2415   \tl_if_empty:NF \l__stex_notation_op_tl {
2416     \cs_set:cpn {
2417       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2418       \l__stex_notation_suffix_str
2419       _cs
2420     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2421   }
2422 }
2423
2424 \stex_debug:nn{symbols}{
2425   Notation~\l__stex_notation_suffix_str
2426   ~for~\l_stex_get_symbol_uri_str^^J
2427   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2428   Argument~precedences:~
2429   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2430   Notation: \cs_meaning:c {
2431     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2432     \l__stex_notation_suffix_str
2433     _cs
2434   }
2435 }
2436
2437 \exp_args:Nx
2438 \stex_do_up_to_module:n {
2439   \seq_put_right:cx {
2440     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2441     _notations
2442   } {
2443     \l__stex_notation_suffix_str
2444   }
2445 }
2446 \exp_args:Ne
2447 \stex_add_to_current_module:n {
2448   \seq_put_right:cn {

```



```

2449     \l_stex_symdecl \l_stex_get_symbol_uri_str
2450     _notations
2451   } { \l__stex_notation_suffix_str }
2452 }
2453
2454 \stex_if_smsmode:F {
2455
2456   % HTML annotations
2457   \stex_if_do_html:T {
2458     \stex_annotate_invisible:nnn { notation }
2459     { \l_stex_get_symbol_uri_str } {
2460       \stex_annotate_invisible:nnn { notationfragment }
2461       { \l__stex_notation_suffix_str }{}
2462       \stex_annotate_invisible:nnn { precedence }
2463       { \l__stex_notation_prec_str }{}
2464
2465       \int_zero:N \l_tmpa_int
2466       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2467       \tl_clear:N \l_tmpa_tl
2468       \int_step_inline:nn { \l__stex_notation_arity_str }{
2469         \int_incr:N \l_tmpa_int
2470         \str_set:Nx \l_tmpb_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_r
2472         \str_if_eq:VnTF \l_tmpb_str a {
2473           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2474             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2475             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2476           } }
2477         }{
2478           \str_if_eq:VnTF \l_tmpb_str B {
2479             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2480               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2481               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2482             } }
2483           }{
2484             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2485               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2486             } }
2487           }
2488         }
2489       }
2490       \stex_annotate_invisible:nnn { notationcomp }{}{
2491         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2492         $ \exp_args:Nno \use:nn { \use:c {
2493           stex_notation_ \l_stex_current_symbol_str
2494           \c_hash_str \l__stex_notation_suffix_str _cs
2495         } } { \l_tmpa_tl } $
2496       }
2497     }
2498   }
2499 }
2500 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2501 \keys_define:nn { stex / setnotation } {
2502   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2503   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2504   unknown .code:n = \str_set:Nx
2505     \l__stex_notation_variant_str \l_keys_key_str
2506 }
2507
2508 \cs_new_protected:Nn \stex_setnotation_args:n {
2509   \str_clear:N \l__stex_notation_lang_str
2510   \str_clear:N \l__stex_notation_variant_str
2511   \keys_set:nn { stex / setnotation } { #1 }
2512 }
2513
2514 \cs_new_protected:Nn \stex_setnotation:n {
2515   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2516     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2517     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2518       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2519     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2520       { \c_hash_str }
2521     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2522       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2523     \exp_args:Nx \stex_add_to_current_module:n {
2524       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2525         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2526       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2527         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2528       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2529         { \c_hash_str }
2530     }
2531     \stex_debug:nn {notations}{
2532       Setting~default~notation~
2533       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2534       #1 \\
2535       \expandafter\meaning\csname
2536         l_stex_symdecl_#1 _notations\endcsname
2537     }
2538   }{
2539     % todo throw error
2540   }
2541 }
2542
2543 \NewDocumentCommand \setnotation {m m} {
2544   \stex_get_symbol:n { #1 }
2545   \stex_setnotation_args:n { #2 }
2546   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2547   \stex_smsmode_do:\ignorespacesandpars
2548 }
2549
2550 \cs_new_protected:Nn \stex_copy_notations:nn {
2551   \stex_debug:nn {notations}{
2552     Copying~notations~from~#2~to~#1\\
2553     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```

```

2554 }
2555 \tl_clear:N \l_tmpa_tl
2556 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2557   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2558 }
2559 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2560   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2561   \edef \l_tmpa_tl {
2562     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2563     \exp_after:wN\exp_after:wN\exp_after:wN {
2564       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2565     }
2566   }
2567   \exp_args:Nx
2568   \stex_do_up_to_module:n {
2569     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2570     \cs_generate_from_arg_count:cNnn {
2571       stex_notation_ #1 \c_hash_str ##1 _cs
2572     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2573       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2574     }
2575   }
2576 }
2577 }
2578
2579 \NewDocumentCommand \copynotation {m m} {
2580   \stex_get_symbol:n { #1 }
2581   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2582   \stex_get_symbol:n { #2 }
2583   \exp_args:Noo
2584   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2585   \exp_args:Nx \stex_add_import_to_current_module:n{
2586     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2587   }
2588   \stex_smsmode_do:\ignorespacesandpars
2589 }
2590

```

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

\symdef

```

2591 \keys_define:nn { stex / symdef } {
2592   name .str_set_x:N = \l_stex_symdecl_name_str ,
2593   local .bool_set:N = \l_stex_symdecl_local_bool ,
2594   args .str_set_x:N = \l_stex_symdecl_args_str ,
2595   type .tl_set:N = \l_stex_symdecl_type_tl ,
2596   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2597   op .tl_set:N = \l_stex_notation_op_tl ,
2598   lang .str_set_x:N = \l_stex_notation_lang_str ,
2599   variant .str_set_x:N = \l_stex_notation_variant_str ,
2600   prec .str_set_x:N = \l_stex_notation_prec_str ,
2601   assoc .choices:nn =
2602     {bin,binl,binr,pre,conj,pwconj}
2603     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}},

```

```

2604   unknown .code:n      = \str_set:Nx
2605       \l__stex_notation_variant_str \l_keys_key_str
2606 }
2607
2608 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2609   \str_clear:N \l_stex_symdecl_name_str
2610   \str_clear:N \l_stex_symdecl_args_str
2611   \str_clear:N \l_stex_symdecl_assoc_type_str
2612   \bool_set_false:N \l_stex_symdecl_local_bool
2613   \tl_clear:N \l_stex_symdecl_type_tl
2614   \tl_clear:N \l_stex_symdecl_definiens_tl
2615   \str_clear:N \l__stex_notation_lang_str
2616   \str_clear:N \l__stex_notation_variant_str
2617   \str_clear:N \l__stex_notation_prec_str
2618   \tl_clear:N \l__stex_notation_op_tl
2619
2620   \keys_set:nn { stex / symdef } { #1 }
2621 }
2622
2623 \NewDocumentCommand \symdef { m O{} } {
2624   \__stex_notation_symdef_args:n { #2 }
2625   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2626   \stex_symdecl_do:n { #1 }
2627   \tl_set:Nn \l_stex_notation_after_do_tl {
2628     \__stex_notation_final:
2629     \stex_smsmode_do:\ignorespacesandpars
2630   }
2631   \str_set:Nx \l_stex_get_symbol_uri_str {
2632     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2633   }
2634   \exp_args:Nx \stex_notation_do:nnnnn
2635     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} {args} }
2636     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} {arity} }
2637     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2638     { \l__stex_notation_prec_str }
2639   }
2640   \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for `\symdef`. This function is documented on page [61](#).)

29.3 Variables

```

2641 <@@=stex_variables>
2642
2643 \keys_define:nn { stex / vardef } {
2644   name .str_set_x:N = \l__stex_variables_name_str ,
2645   args .str_set_x:N = \l__stex_variables_args_str ,
2646   type .tl_set:N    = \l__stex_variables_type_tl ,
2647   def  .tl_set:N    = \l__stex_variables_def_tl ,
2648   op   .tl_set:N    = \l__stex_variables_op_tl ,
2649   prec .str_set_x:N = \l__stex_variables_prec_str ,
2650   assoc .choices:nn =
2651     {bin,binl,binr,pre,conj,pwconj}
2652     {\str_set:Nx \l__stex_variables_assoc_type_str {\l_keys_choice_tl}},

```

```

2653 bind .choices:nn =
2654 {forall,exists}
2655 {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2656 }
2657
2658 \cs_new_protected:Nn \__stex_variables_args:n {
2659 \str_clear:N \l__stex_variables_name_str
2660 \str_clear:N \l__stex_variables_args_str
2661 \str_clear:N \l__stex_variables_prec_str
2662 \str_clear:N \l__stex_variables_assoctype_str
2663 \str_clear:N \l__stex_variables_bind_str
2664 \tl_clear:N \l__stex_variables_type_tl
2665 \tl_clear:N \l__stex_variables_def_tl
2666 \tl_clear:N \l__stex_variables_op_tl
2667
2668 \keys_set:nn { stex / vardef } { #1 }
2669 }
2670
2671 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2672 \__stex_variables_args:n {#2}
2673 \str_if_empty:NT \l__stex_variables_name_str {
2674 \str_set:Nx \l__stex_variables_name_str { #1 }
2675 }
2676 \prop_clear:N \l_tmpa_prop
2677 \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2678
2679 \int_zero:N \l_tmpb_int
2680 \bool_set_true:N \l_tmpa_bool
2681 \str_map_inline:Nn \l__stex_variables_args_str {
2682 \token_case_meaning:NnF ##1 {
2683 0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2684 {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2685 {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2686 {\tl_to_str:n a} {
2687 \bool_set_false:N \l_tmpa_bool
2688 \int_incr:N \l_tmpb_int
2689 }
2690 {\tl_to_str:n B} {
2691 \bool_set_false:N \l_tmpa_bool
2692 \int_incr:N \l_tmpb_int
2693 }
2694 }{
2695 \msg_error:nnxx{stex}{error/wrongargs}{
2696 variable~\l__stex_variables_name_str
2697 }{##1}
2698 }
2699 }
2700 \bool_if:NTF \l_tmpa_bool {
2701 % possibly numeric
2702 \str_if_empty:NTF \l__stex_variables_args_str {
2703 \prop_put:Nnn \l_tmpa_prop { args } {}
2704 \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2705 }{
2706 \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }

```

```

2707 \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2708 \str_clear:N \l_tmpa_str
2709 \int_step_inline:nn \l_tmpa_int {
2710   \str_put_right:Nn \l_tmpa_str i
2711 }
2712 \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2713 \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2714 }
2715 } {
2716   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2717   \prop_put:Nnx \l_tmpa_prop { arity }
2718   { \str_count:N \l__stex_variables_args_str }
2719 }
2720 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2721 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2722
2723 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2724
2725 \tl_if_empty:NF \l__stex_variables_op_tl {
2726   \cs_set:cpx {
2727     stex_var_op_notation_\l__stex_variables_name_str_cs
2728   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2729 }
2730
2731 \tl_set:Nn \l_stex_notation_after_do_tl {
2732   \exp_args:Nne \use:nn {
2733     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2734     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2735   } {{
2736     \exp_after:wN \exp_after:wN \exp_after:wN
2737     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2738     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2739   }}
2740 \stex_if_do_html:T {
2741   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2742     \stex_annotate_invisible:nnn { precedence }
2743     { \l__stex_variables_prec_str }{}
2744   \tl_if_empty:NF \l__stex_variables_type_tl { \stex_annotate_invisible:nnn{type}{\l__stex_variables_type_tl}{
2745     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2746     \stex_annotate_invisible:nnn{macroname}{#1}{
2747   \tl_if_empty:NF \l__stex_variables_def_tl {
2748     \stex_annotate_invisible:nnn{definiens}{
2749       {\l__stex_variables_def_tl$}
2750   }
2751   \str_if_empty:NF \l__stex_variables_assoc_type_str {
2752     \stex_annotate_invisible:nnn{assoc_type}{\l__stex_variables_assoc_type_str}{
2753   }
2754   \int_zero:N \l_tmpa_int
2755   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2756   \tl_clear:N \l_tmpa_tl
2757   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2758     \int_incr:N \l_tmpa_int
2759     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2760     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables

```

```

2761 \str_if_eq:VnTF \l_tmpb_str a {
2762 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2763 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2764 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2765 } }
2766 }{
2767 \str_if_eq:VnTF \l_tmpb_str B {
2768 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2769 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2770 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2771 } }
2772 }{
2773 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2774 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2775 } }
2776 }
2777 }
2778 }
2779 \stex_annotate_invisible:nnn { notationcomp }{}{
2780 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2781 $ \exp_args:Nno \use:nn { \use:c {
2782 stex_var_notation_\l__stex_variables_name_str _cs
2783 } } { \l_tmpa_tl } $
2784 }
2785 }
2786 }\ignorespacesandpars
2787 }
2788
2789 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2790 }
2791
2792 \cs_new:Nn \_stex_reset:N {
2793 \tl_if_exist:NTF #1 {
2794 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2795 }{
2796 \let \exp_not:N #1 \exp_not:N \undefined
2797 }
2798 }
2799
2800 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2801 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2802 \exp_args:Nnx \use:nn {
2803 % TODO
2804 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2805 #2
2806 }
2807 }{
2808 \_stex_reset:N \varnot
2809 \_stex_reset:N \vartype
2810 \_stex_reset:N \vardefi
2811 }
2812 }
2813
2814 \NewDocumentCommand \vardef { s } {

```

```

2815 \IfBooleanTF#1 {
2816   \__stex_variables_do_complex:nn
2817 }{
2818   \__stex_variables_do_simple:nnn
2819 }
2820 }
2821
2822 \NewDocumentCommand \svar { 0{} m }{
2823   \tl_if_empty:nTF {#1}{
2824     \str_set:Nn \l_tmpa_str { #2 }
2825   }{
2826     \str_set:Nn \l_tmpa_str { #1 }
2827   }
2828   \_stex_term_omv:nn {
2829     var://\l_tmpa_str
2830   }{
2831     \exp_args:Nnx \use:nn {
2832       \def\comp{\_varcomp}
2833       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2834       \comp{ #2 }
2835     }{
2836       \_stex_reset:N \comp
2837       \_stex_reset:N \l_stex_current_symbol_str
2838     }
2839   }
2840 }
2841
2842
2843
2844 \keys_define:nn { stex / varseq } {
2845   name .str_set_x:N = \l__stex_variables_name_str ,
2846   args .int_set:N = \l__stex_variables_args_int ,
2847   type .tl_set:N = \l__stex_variables_type_tl ,
2848   mid .tl_set:N = \l__stex_variables_mid_tl ,
2849   bind .choices:nn =
2850     {forall,exists}
2851     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2852 }
2853
2854 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2855   \str_clear:N \l__stex_variables_name_str
2856   \int_set:Nn \l__stex_variables_args_int 1
2857   \tl_clear:N \l__stex_variables_type_tl
2858   \str_clear:N \l__stex_variables_bind_str
2859
2860   \keys_set:nn { stex / varseq } { #1 }
2861 }
2862
2863 \NewDocumentCommand \varseq {m 0{} m m m}{
2864   \__stex_variables_seq_args:n { #2 }
2865   \str_if_empty:NT \l__stex_variables_name_str {
2866     \str_set:Nx \l__stex_variables_name_str { #1 }
2867   }
2868   \prop_clear:N \l_tmpa_prop

```



```

2869 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2870
2871 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2872 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2873   \msg_error:nnxx{stex}{error/seqlength}
2874   {\int_use:N \l__stex_variables_args_int}
2875   {\seq_count:N \l_tmpa_seq}
2876 }
2877 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2878 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2879   \msg_error:nnxx{stex}{error/seqlength}
2880   {\int_use:N \l__stex_variables_args_int}
2881   {\seq_count:N \l_tmpb_seq}
2882 }
2883 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2884 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2885
2886 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2887   \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
2888
2889 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2890 \int_step_inline:nn \l__stex_variables_args_int {
2891   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2892 }
2893 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2894 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2895 \tl_if_empty:NF \l__stex_variables_mid_tl {
2896   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2897   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2898 }
2899 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2900 \int_step_inline:nn \l__stex_variables_args_int {
2901   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2902 }
2903 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2904 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2905
2906
2907 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2908
2909 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2910
2911 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2912
2913 \int_step_inline:nn \l__stex_variables_args_int {
2914   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2915     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2916   }}
2917 }
2918
2919 \tl_set:Nx \l_tmpa_tl {
2920   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{\l_tmpa_tl}
2921   \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2922 }

```

```

2923 }
2924
2925 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2926
2927 \exp_args:Nno \use:nn {
2928 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2929 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
2930
2931 \stex_debug:nn{sequences}{New~Sequence:~
2932 \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2933 \prop_to_keyval:N \l_tmpa_prop
2934 }
2935
2936 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2937 \ignorespacesandpars
2938 }
2939
2940 \</package>

```

Chapter 30

STEX -Terms Implementation

```
2941 <*package>
2942
2943 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2944
2945 <@@=stex_terms>
2946
2947 Warnings and error messages
2948 \msg_new:nnn{stex}{error/nonotation}{
2949   Symbol~#1~invoked,~but~has~no~notation~#2!
2950 }
2951 \msg_new:nnn{stex}{error/notationarg}{
2952   Error~in~parsing~notation~#1
2953 }
2954 \msg_new:nnn{stex}{error/noop}{
2955   Symbol~#1~has~no~operator~notation~for~notation~#2
2956 }
2957 \msg_new:nnn{stex}{error/notallowed}{
2958   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
2959 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
2959
2960
2961 \bool_new:N \l_stex_allow_semantic_bool
2962 \bool_set_true:N \l_stex_allow_semantic_bool
2963
2964 \cs_new_protected:Nn \stex_invoke_symbol:n {
2965   \bool_if:NTF \l_stex_allow_semantic_bool {
2966     \str_if_eq:eeF {
2967       \prop_item:cn {
2968         l_stex_symdecl_#1_prop
2969       }{ deprecate }
2970     }
```

```

2970   }{}{
2971     \msg_warning:nxxx{stex}{warning/deprecated}{
2972       Symbol~#1
2973     }{
2974       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
2975     }
2976   }
2977   \if_mode_math:
2978     \exp_after:wN \__stex_terms_invoke_math:n
2979   \else:
2980     \exp_after:wN \__stex_terms_invoke_text:n
2981   \fi: { #1 }
2982 }{
2983   \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
2984 }
2985 }
2986
2987 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2988   \peek_charcode_remove:NTF ! {
2989     \__stex_terms_invoke_op_custom:nn {#1}
2990   }{
2991     \__stex_terms_invoke_custom:nn {#1}
2992   }
2993 }
2994
2995 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2996   \peek_charcode_remove:NTF ! {
2997     % operator
2998     \peek_charcode_remove:NTF * {
2999       % custom op
3000       \__stex_terms_invoke_op_custom:nn {#1}
3001     }{
3002       % op notation
3003       \peek_charcode:NTF [ {
3004         \__stex_terms_invoke_op_notation:nw {#1}
3005       }{
3006         \__stex_terms_invoke_op_notation:nw {#1}[]
3007       }
3008     }
3009   }{
3010     \peek_charcode_remove:NTF * {
3011       \__stex_terms_invoke_custom:nn {#1}
3012       % custom
3013     }{
3014       % normal
3015       \peek_charcode:NTF [ {
3016         \__stex_terms_invoke_notation:nw {#1}
3017       }{
3018         \__stex_terms_invoke_notation:nw {#1}[]
3019       }
3020     }
3021   }
3022 }
3023

```

```

3024
3025 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3026   \exp_args:Nnx \use:nn {
3027     \def\comp{\_comp}
3028     \str_set:Nn \l_stex_current_symbol_str { #1 }
3029     \bool_set_false:N \l_stex_allow_semantic_bool
3030     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3031       \comp{ #2 }
3032     }
3033   }{
3034     \stex_reset:N \comp
3035     \stex_reset:N \l_stex_current_symbol_str
3036     \bool_set_true:N \l_stex_allow_semantic_bool
3037   }
3038 }
3039
3040 \keys_define:nn { stex / terms } {
3041   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3042   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3043   unknown .code:n = \str_set:Nx
3044     \l_stex_notation_variant_str \l_keys_key_str
3045 }
3046
3047 \cs_new_protected:Nn \__stex_terms_args:n {
3048   \str_clear:N \l_stex_notation_lang_str
3049   \str_clear:N \l_stex_notation_variant_str
3050
3051   \keys_set:nn { stex / terms } { #1 }
3052 }
3053
3054 \cs_new_protected:Nn \stex_find_notation:nn {
3055   \__stex_terms_args:n { #2 }
3056   \seq_if_empty:cTF {
3057     l_stex_symdecl_ #1 _notations
3058   } {
3059     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3060   } {
3061     \bool_lazy_all:nTF {
3062       {\str_if_empty_p:N \l_stex_notation_variant_str}
3063       {\str_if_empty_p:N \l_stex_notation_lang_str}
3064     }{
3065       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3066     }{
3067       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3068         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3069       }{
3070         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3071       }{
3072         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3073           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3074         }
3075       }
3076     }
3077   }

```

```

3078 }
3079
3080 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3081   \exp_args:Nnx \use:nn {
3082     \def\comp{\_comp}
3083     \str_set:Nn \l_stex_current_symbol_str { #1 }
3084     \stex_find_notation:nn { #1 }{ #2 }
3085     \bool_set_false:N \l_stex_allow_semantic_bool
3086     \cs_if_exist:cTF {
3087       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3088     }{
3089       \_stex_term_oms:nnn {
3090         #1 \c_hash_str \l_stex_notation_variant_str
3091       }{ #1 }{
3092         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3093       }
3094     }{
3095       \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3096         \cs_if_exist:cTF {
3097           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3098         }{
3099           \tl_set:Nx \stex_symbol_after_invokation_tl {
3100             \_stex_reset:N \comp
3101             \_stex_reset:N \stex_symbol_after_invokation_tl
3102             \_stex_reset:N \l_stex_current_symbol_str
3103             \bool_set_true:N \l_stex_allow_semantic_bool
3104           }
3105           \def\comp{\_comp}
3106           \str_set:Nn \l_stex_current_symbol_str { #1 }
3107           \bool_set_false:N \l_stex_allow_semantic_bool
3108           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3109         }{
3110           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3111             ~\l_stex_notation_variant_str
3112           }
3113         }
3114       }{
3115         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3116       }
3117     }
3118   }{
3119     \_stex_reset:N \comp
3120     \_stex_reset:N \l_stex_current_symbol_str
3121     \bool_set_true:N \l_stex_allow_semantic_bool
3122   }
3123 }
3124
3125 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3126   \stex_find_notation:nn { #1 }{ #2 }
3127   \cs_if_exist:cTF {
3128     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3129   }{
3130     \tl_set:Nx \stex_symbol_after_invokation_tl {
3131       \_stex_reset:N \comp

```

```

3132     \stex_reset:N \stex_symbol_after_invokation_tl
3133     \stex_reset:N \l_stex_current_symbol_str
3134     \bool_set_true:N \l_stex_allow_semantic_bool
3135   }
3136   \def\comp{\_comp}
3137   \str_set:Nn \l_stex_current_symbol_str { #1 }
3138   \bool_set_false:N \l_stex_allow_semantic_bool
3139   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3140 }{
3141   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3142     ~\l_stex_notation_variant_str
3143   }
3144 }
3145 }
3146
3147 \prop_new:N \l__stex_terms_custom_args_prop
3148
3149 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3150   \exp_args:Nnx \use:nn {
3151     \bool_set_false:N \l_stex_allow_semantic_bool
3152     \def\comp{\_comp}
3153     \str_set:Nn \l_stex_current_symbol_str { #1 }
3154     \prop_clear:N \l__stex_terms_custom_args_prop
3155     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3156     \prop_get:cnN {
3157       l_stex_symdecl_#1 _prop
3158     }{ args } \l_tmpa_str
3159     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3160     \tl_set:Nn \arg { \__stex_terms_arg: }
3161     \str_if_empty:NTF \l_tmpa_str {
3162       \stex_term_oms:nnn {#1}{#1}{#2}
3163     }{
3164       \str_if_in:NnTF \l_tmpa_str b {
3165         \stex_term_ombind:nnn {#1}{#1}{#2}
3166       }{
3167         \str_if_in:NnTF \l_tmpa_str B {
3168           \stex_term_ombind:nnn {#1}{#1}{#2}
3169         }{
3170           \stex_term_oma:nnn {#1}{#1}{#2}
3171         }
3172       }
3173     }
3174     % TODO check that all arguments exist
3175   }{
3176     \stex_reset:N \l_stex_current_symbol_str
3177     \stex_reset:N \arg
3178     \stex_reset:N \comp
3179     \stex_reset:N \l__stex_terms_custom_args_prop
3180     \bool_set_true:N \l_stex_allow_semantic_bool
3181   }
3182 }
3183
3184 \NewDocumentCommand \__stex_terms_arg: { s O{} m }{
3185   \tl_if_empty:nTF {#2}{

```

```

3186 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3187 \bool_set_true:N \l_tmpa_bool
3188 \bool_do_while:Nn \l_tmpa_bool {
3189   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3190   \int_incr:N \l_tmpa_int
3191   }{
3192     \bool_set_false:N \l_tmpa_bool
3193   }
3194 }
3195 ){
3196   \int_set:Nn \l_tmpa_int { #2 }
3197   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3198     % TODO throw error
3199   }
3200 }
3201 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3202 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3203   % TODO throw error
3204 }
3205 \bool_set_true:N \l_stex_allow_semantic_bool
3206 \IfBooleanTF#1{
3207   \stex_annotate_invisible:n {
3208     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3209   }
3210 }{
3211   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3212 }
3213 \bool_set_false:N \l_stex_allow_semantic_bool
3214 }
3215
3216
3217 \cs_new_protected:Nn \_stex_term_arg:nn {
3218   \bool_set_true:N \l_stex_allow_semantic_bool
3219   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3220   \bool_set_false:N \l_stex_allow_semantic_bool
3221 }
3222
3223 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3224   \exp_args:Nnx \use:nn
3225   { \int_set:Nn \l__stex_terms_downprec { #2 }
3226     \_stex_term_arg:nn { #1 }{ #3 }
3227   }
3228   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3229 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3230 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3231   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3232   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3233   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3234     \expandafter\if\expandafter\relax\noexpand#3
3235     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```



```

3236 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3237 }{
3238 \__stex_terms_math_assoc_arg_simple:n{#3}
3239 }
3240 }
3241
3242 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3243 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3244 \str_if_empty:NTF \l_tmpa_str {
3245 \exp_args:Nx \cs_if_eq:NNTF {
3246 \tl_head:N #1
3247 } \stex_invoke_sequence:n {
3248 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3249 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3250 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3251 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3252 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3253 \exp_not:n{\exp_args:Nnx \use:nn} {
3254 \exp_not:n {
3255 \def\comp{\_varcomp}
3256 \str_set:Nn \l_stex_current_symbol_str
3257 } {varseq://\l_tmpa_str}
3258 \exp_not:n{ ##1 }
3259 }{
3260 \exp_not:n {
3261 \_stex_reset:N \comp
3262 \_stex_reset:N \l_stex_current_symbol_str
3263 }
3264 }
3265 }}}
3266 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3267 \seq_reverse:N \l_tmpa_seq
3268 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3269 \seq_map_inline:Nn \l_tmpa_seq {
3270 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3271 \exp_args:Nno
3272 \l_tmpa_cs { ##1 } \l_tmpa_tl
3273 }
3274 }
3275 \tl_set:Nx \l_tmpa_tl {
3276 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3277 \exp_args:No \exp_not:n \l_tmpa_tl
3278 }
3279 }
3280 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3281 }{
3282 \__stex_terms_math_assoc_arg_simple:n { #1 }
3283 }
3284 } {
3285 \__stex_terms_math_assoc_arg_simple:n { #1 }
3286 }
3287
3288 }
3289

```

```

3290 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3291   \clist_set:Nn \l_tmpa_clist{ #1 }
3292   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3293     \tl_set:Nn \l_tmpa_tl { #1 }
3294   }{
3295     \clist_reverse:N \l_tmpa_clist
3296     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3297
3298     \clist_map_inline:Nn \l_tmpa_clist {
3299       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3300         \exp_args:Nno
3301         \l_tmpa_cs { ##1 } \l_tmpa_tl
3302       }
3303     }
3304   }
3305   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3306 }

```

(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
3307 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3308 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3309 \int_new:N \l__stex_terms_downprec
3310 \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
3311 \tl_set:Nn \l__stex_terms_left_bracket_str (
3312 \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

```

3313 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3314   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3315     \bool_set_false:N \l__stex_terms_brackets_done_bool
3316     #2
3317   } {
3318     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3319       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3320         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3321         \dobrackets { #2 }
3322       }
3323     }{ #2 }
3324   }
3325 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

3326 \bool_new:N \l__stex_terms_brackets_done_bool
3327 %\RequirePackage{scalerel}
3328 \cs_new_protected:Npn \dobrackets #1 {
3329   %\ThisStyle{\if D\m@switch
3330   %   \exp_args:Nnx \use:nn
3331   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3332   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3333   % \else
3334   \exp_args:Nnx \use:nn
3335   {
3336     \bool_set_true:N \l__stex_terms_brackets_done_bool
3337     \int_set:Nn \l__stex_terms_downprec \infpref
3338     \l__stex_terms_left_bracket_str
3339     #1
3340   }
3341   {
3342     \bool_set_false:N \l__stex_terms_brackets_done_bool
3343     \l__stex_terms_right_bracket_str
3344     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3345   }
3346   %\fi}
3347 }
```

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

`\withbrackets`

```

3348 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3349   \exp_args:Nnx \use:nn
3350   {
3351     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3352     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3353     #3
3354   }
3355   {
3356     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3357     {\l__stex_terms_left_bracket_str}
3358     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3359     {\l__stex_terms_right_bracket_str}
3360   }
3361 }
```

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

`\STEXinvisible`

```

3362 \cs_new_protected:Npn \STEXinvisible #1 {
3363   \stex_annotate_invisible:n { #1 }
3364 }
```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
3365 \cs_new_protected:Nn \_stex_term_oms:nnn {
3366   \stex_annotate:nnn{ OMID }{ #2 }{
3367     \stex_highlight_term:nn { #1 } { #3 }
3368   }
3369 }
3370
3371 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3372   \__stex_terms_maybe_brackets:nn { #3 }{
3373     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3374   }
3375 }
```

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

`_stex_term_math_omv:nn`

```
3376 \cs_new_protected:Nn \_stex_term_omv:nn {
3377   \stex_annotate:nnn{ OMV }{ #1 }{
3378     \stex_highlight_term:nn { #1 } { #2 }
3379   }
3380 }
```

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

`_stex_term_math_oma:nnnn`

```
3381 \cs_new_protected:Nn \_stex_term_oma:nnn {
3382   \stex_annotate:nnn{ OMA }{ #2 }{
3383     \stex_highlight_term:nn { #1 } { #3 }
3384   }
3385 }
3386
3387 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3388   \__stex_terms_maybe_brackets:nn { #3 }{
3389     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3390   }
3391 }
```

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

`_stex_term_math_omb:nnnn`

```
3392 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3393   \stex_annotate:nnn{ OMBIND }{ #2 }{
3394     \stex_highlight_term:nn { #1 } { #3 }
3395   }
3396 }
3397
3398 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3399   \__stex_terms_maybe_brackets:nn { #3 }{
3400     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3401   }
3402 }
```

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

```

\symref
\symname
3403 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3404
3405 \keys_define:nn { stex / symname } {
3406   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3407   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3408   root     .tl_set_x:N = \l__stex_terms_root_tl
3409 }
3410
3411 \cs_new_protected:Nn \stex_symname_args:n {
3412   \tl_clear:N \l__stex_terms_post_tl
3413   \tl_clear:N \l__stex_terms_pre_tl
3414   \tl_clear:N \l__stex_terms_root_str
3415   \keys_set:nn { stex / symname } { #1 }
3416 }
3417
3418 \NewDocumentCommand \symref { m m }{
3419   \let\compemph_uri_prev:\compemph@uri
3420   \let\compemph@uri\symrefemph@uri
3421   \STEXsymbol{#1}!\{ #2 }
3422   \let\compemph@uri\compemph_uri_prev:
3423 }
3424
3425 \NewDocumentCommand \synonym { 0{} m m }{
3426   \stex_symname_args:n { #1 }
3427   \let\compemph_uri_prev:\compemph@uri
3428   \let\compemph@uri\symrefemph@uri
3429   % TODO
3430   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3431   \let\compemph@uri\compemph_uri_prev:
3432 }
3433
3434 \NewDocumentCommand \symname { 0{} m }{
3435   \stex_symname_args:n { #1 }
3436   \stex_get_symbol:n { #2 }
3437   \str_set:Nx \l_tmpa_str {
3438     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3439   }
3440   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3441
3442   \let\compemph_uri_prev:\compemph@uri
3443   \let\compemph@uri\symrefemph@uri
3444   \exp_args:NNx \use:nn
3445   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3446     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3447   } }
3448   \let\compemph@uri\compemph_uri_prev:
3449 }
3450
3451 \NewDocumentCommand \Symname { 0{} m }{
3452   \stex_symname_args:n { #1 }
3453   \stex_get_symbol:n { #2 }
3454   \str_set:Nx \l_tmpa_str {
3455     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3456 }
3457 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3458 \let\compemph_uri_prev:\compemph@uri
3459 \let\compemph@uri\symrefemph@uri
3460 \exp_args:NNx \use:nn
3461 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3462   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3463   \l__stex_terms_post_tl
3464 } }
3465 \let\compemph@uri\compemph_uri_prev:
3466 }

```

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

30.3 Notation Components

```

3467 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3468 \cs_new_protected:Nn \stex_highlight_term:nn {
3469   #2
3470 }
3471
3472 \cs_new_protected:Nn \stex_unhighlight_term:n {
3473   % \latexml_if:TF {
3474   %   #1
3475   % } {
3476   %   \rustex_if:TF {
3477   %     #1
3478   %   } {
3479     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3480   % }
3481   % }
3482 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3483 \cs_new_protected:Npn \_comp #1 {
3484   \str_if_empty:NF \l_stex_current_symbol_str {
3485     \rustex_if:TF {
3486       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3487     }{
3488       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3489     }
3490   }
3491 }
3492
3493 \cs_new_protected:Npn \_varcomp #1 {
3494   \str_if_empty:NF \l_stex_current_symbol_str {
3495     \rustex_if:TF {
3496       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3497     }{
3498       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3499     }
3500   }
3501 }
3502
3503 \def\comp{\_comp}
3504
3505 \cs_new_protected:Npn \compemph@uri #1 #2 {
3506   \compemph{ #1 }
3507 }
3508
3509
3510 \cs_new_protected:Npn \compemph #1 {
3511   #1
3512 }
3513
3514 \cs_new_protected:Npn \defemph@uri #1 #2 {
3515   \defemph{#1}
3516 }
3517
3518 \cs_new_protected:Npn \defemph #1 {
3519   \textbf{#1}
3520 }
3521
3522 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3523   \symrefemph{#1}
3524 }
3525
3526 \cs_new_protected:Npn \symrefemph #1 {
3527   \textbf{#1}
3528 }
3529
3530 \cs_new_protected:Npn \varemp@uri #1 #2 {
3531   \varemp{#1}
3532 }
3533
3534 \cs_new_protected:Npn \varemp #1 {
3535   #1
3536 }

```

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

\ellipses

```

3537 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3538 \bool_new:N \l_stex_inarray_bool
\parrayline 3539 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 3540 \NewDocumentCommand \parray { m m } {
\parraycell 3541   \begingroup
3542   \bool_set_true:N \l_stex_inarray_bool
3543   \begin{array}{#1}
3544     #2
3545   \end{array}

```

```

3546 \endgroup
3547 }
3548
3549 \NewDocumentCommand \prmatrix { m } {
3550 \begingroup
3551 \bool_set_true:N \l_stex_inarray_bool
3552 \begin{matrix}
3553 #1
3554 \end{matrix}
3555 \endgroup
3556 }
3557
3558 \def \maybepline {
3559 \bool_if:NT \l_stex_inarray_bool {\hline}
3560 }
3561
3562 \def \parrayline #1 #2 {
3563 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3564 }
3565
3566 \def \pmrow #1 { \parrayline{}{ #1 } }
3567
3568 \def \parraylineh #1 #2 {
3569 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3570 }
3571
3572 \def \parraycell #1 {
3573 #1 \bool_if:NT \l_stex_inarray_bool {&}
3574 }

```

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

30.4 Variables

```

3575 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3576 \cs_new_protected:Nn \stex_invoke_variable:n {
3577 \if_mode_math:
3578 \exp_after:wN \__stex_variables_invoke_math:n
3579 \else:
3580 \exp_after:wN \__stex_variables_invoke_text:n
3581 \fi: {#1}
3582 }
3583
3584 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3585 %TODO
3586 }
3587
3588
3589 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3590 \peek_charcode_remove:NTF ! {
3591 \peek_charcode_remove:NTF ! {
3592 \peek_charcode:NTF [ {

```



```

3593     \__stex_variables_invoke_op_custom:nw
3594   }{
3595     % TODO throw error
3596   }
3597 }{
3598   \__stex_variables_invoke_op:n { #1 }
3599 }
3600 }{
3601   \peek_charcode_remove:NTF * {
3602     \__stex_variables_invoke_text:n { #1 }
3603   }{
3604     \__stex_variables_invoke_math_ii:n { #1 }
3605   }
3606 }
3607 }
3608
3609 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3610   \cs_if_exist:cTF {
3611     stex_var_op_notation_ #1 _cs
3612   }{
3613     \exp_args:Nnx \use:nn {
3614       \def\comp{\_varcomp}
3615       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3616       \_stex_term_omv:nn { var://#1 }{
3617         \use:c{stex_var_op_notation_ #1 _cs }
3618       }
3619     }{
3620       \_stex_reset:N \comp
3621       \_stex_reset:N \l_stex_current_symbol_str
3622     }
3623   }{
3624     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3625       \__stex_variables_invoke_math_ii:n {#1}
3626     }{
3627       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3628     }
3629   }
3630 }
3631
3632 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3633   \cs_if_exist:cTF {
3634     stex_var_notation_#1_cs
3635   }{
3636     \tl_set:Nx \stex_symbol_after_invokation_tl {
3637       \_stex_reset:N \comp
3638       \_stex_reset:N \stex_symbol_after_invokation_tl
3639       \_stex_reset:N \l_stex_current_symbol_str
3640       \bool_set_true:N \l_stex_allow_semantic_bool
3641     }
3642     \def\comp{\_varcomp}
3643     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3644     \bool_set_false:N \l_stex_allow_semantic_bool
3645     \use:c{stex_var_notation_#1_cs}
3646   }{

```

```

3647 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3648 }
3649 }

```

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

30.5 Sequences

```

3650 <@@=stex_sequences>
3651
3652 \cs_new_protected:Nn \stex_invoke_sequence:n {
3653   \peek_charcode_remove:NTF ! {
3654     \stex_term_omv:nn {varseq://#1}{
3655       \exp_args:Nnx \use:nn {
3656         \def\comp{\_varcomp}
3657         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3658         \prop_item:cn{stex_varseq_#1_prop}{notation}
3659       }{
3660         \stex_reset:N \comp
3661         \stex_reset:N \l_stex_current_symbol_str
3662       }
3663     }
3664   }{
3665     \bool_set_false:N \l_stex_allow_semantic_bool
3666     \def\comp{\_varcomp}
3667     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3668     \tl_set:Nx \stex_symbol_after_invokation_tl {
3669       \stex_reset:N \comp
3670       \stex_reset:N \stex_symbol_after_invokation_tl
3671       \stex_reset:N \l_stex_current_symbol_str
3672       \bool_set_true:N \l_stex_allow_semantic_bool
3673     }
3674     \use:c { stex_varseq_#1_cs }
3675   }
3676 }
3677 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3678 ⟨*package⟩
3679
3680 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3681
3682     Warnings and error messages
3683 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3684     Symbol~#1~can~not~be~assigned~in~copymodule~#2
3685 }
3686 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3687     Symbol~#1~not~assigned~in~interpretmodule~#2
3688 }
3689 \msg_new:nnn{stex}{error/unknownstructure}{
3690     No~structure~#1~found!
3691 }
3692
3693 \msg_new:nnn{stex}{error/unknownfield}{
3694     No~field~#1~in~instance~#2~found!\#3
3695 }
3696
3697 \msg_new:nnn{stex}{error/keyval}{
3698     Invalid~key=value~pair:#1
3699 }
3700 \msg_new:nnn{stex}{error/instantiate/missing}{
3701     Assignments~missing~in~instantiate:~#1
3702 }
3703 \msg_new:nnn{stex}{error/incompatible}{
3704     Incompatible~signature:~#1~(#2)~and~#3~(#4)
3705 }
3706
```

31.1 Imports with modification

```

3707 <@@=stex_copymodule>
3708 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3709   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3710     \tl_set:Nn \l_tmpa_tl { #1 }
3711     \__stex_copymodule_get_symbol_from_cs:
3712   }{
3713     % argument is a string
3714     % is it a command name?
3715     \cs_if_exist:cTF { #1 }{
3716       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3717       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3718       \str_if_empty:NTF \l_tmpa_str {
3719         \exp_args:Nx \cs_if_eq:NNTF {
3720           \tl_head:N \l_tmpa_tl
3721         } \stex_invoke_symbol:n {
3722           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3723         }{
3724           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3725         }
3726       } {
3727         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3728       }
3729     }{
3730       % argument is not a command name
3731       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3732       % \l_stex_all_symbols_seq
3733     }
3734   }
3735 }
3736
3737 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3738   \str_set:Nn \l_tmpa_str { #1 }
3739   \bool_set_false:N \l_tmpa_bool
3740   \bool_if:NF \l_tmpa_bool {
3741     \tl_set:Nn \l_tmpa_tl {
3742       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3743     }
3744     \str_set:Nn \l_tmpa_str { #1 }
3745     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3746     \seq_map_inline:Nn #2 {
3747       \str_set:Nn \l_tmpb_str { ##1 }
3748       \str_if_eq:eeT { \l_tmpa_str } {
3749         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3750       } {
3751         \seq_map_break:n {
3752           \tl_set:Nn \l_tmpa_tl {
3753             \str_set:Nn \l_stex_get_symbol_uri_str {
3754               ##1
3755             }
3756           }
3757         }
3758       }

```

```

3759     }
3760     \l_tmpa_tl
3761   }
3762 }
3763
3764 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3765   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3766     { \tl_tail:N \l_tmpa_tl }
3767   \tl_if_single:NTF \l_tmpa_tl {
3768     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3769       \exp_after:wN \str_set:Nn \exp_after:wN
3770         \l_stex_get_symbol_uri_str \l_tmpa_tl
3771       \__stex_copymodule_get_symbol_check:n { #1 }
3772     }{
3773       % TODO
3774       % tail is not a single group
3775     }
3776   }{
3777     % TODO
3778     % tail is not a single group
3779   }
3780 }
3781
3782 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3783   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3784     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3785       :~\seq_use:Nn #1 {,~}
3786     }
3787   }
3788 }
3789
3790 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3791   \stex_import_module_uri:nn { #1 } { #2 }
3792   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3793   \stex_import_require_module:nnnn
3794     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3795     { \l_stex_import_path_str } { \l_stex_import_name_str }
3796   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3797   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3798   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3799   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3800     \seq_map_inline:cn {c_stex_module_###_constants}{
3801       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3802         ##1 ? ####1
3803       }
3804     }
3805   }
3806   \seq_clear:N \l_tmpa_seq
3807   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3808     name      = \l_stex_current_copymodule_name_str ,
3809     module    = \l_stex_current_module_str ,
3810     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3811     includes  = \l_tmpa_seq ,
3812     fields    = \l_tmpa_seq

```

```

3813 }
3814 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3815   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3816   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3817 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3818 \stex_if_smsmode:F {
3819   \begin{stex_annotate_env} {#4} {
3820     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3821   }
3822   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3823 }
3824 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3825 \bool_set_false:N \_stex_html_do_output_bool
3826 }
3827 \cs_new_protected:Nn \stex_copymodule_end:n {
3828   \def \l_tmpa_cs ##1 ##2 {#1}
3829   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3830   \tl_clear:N \l_tmpa_tl
3831   \tl_clear:N \l_tmpb_tl
3832   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3833   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3834     \seq_map_inline:cn {c_stex_module_##1_constants}{
3835       \tl_clear:N \l_tmpc_tl
3836       \l_tmpa_cs{##1}{####1}
3837       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3838         \tl_put_right:Nx \l_tmpa_tl {
3839           \prop_set_from_keyval:cn {
3840             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3841           }{
3842             \exp_after:wN \prop_to_keyval:N \csname
3843               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3844             \endcsname
3845           }
3846           \seq_clear:c {
3847             l_stex_symdecl_
3848             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
3849             _notations
3850           }
3851         }
3852         \tl_put_right:Nx \l_tmpc_tl {
3853           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
3854           \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1
3855         }
3856         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
3857         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3858           \tl_put_right:Nx \l_tmpc_tl {
3859             \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1
3860           }
3861           \tl_put_right:Nx \l_tmpa_tl {
3862             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3863               \stex_invoke_symbol:n {
3864                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
3865             }
3866           }

```

```

3867     }
3868   }
3869 }{
3870   \tl_put_right:Nx \l_tmpc_tl {
3871     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3872   }
3873   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3874   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3875   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3876   \tl_put_right:Nx \l_tmpa_tl {
3877     \prop_set_from_keyval:cn {
3878       l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3879     }{
3880       \prop_to_keyval:N \l_tmpa_prop
3881     }
3882     \seq_clear:c {
3883       l_stex_symdecl_
3884       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3885       _notations
3886     }
3887   }
3888   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3889   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3890     \tl_put_right:Nx \l_tmpc_tl {
3891       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3892     }
3893     \tl_put_right:Nx \l_tmpa_tl {
3894       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3895         \stex_invoke_symbol:n {
3896           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3897         }
3898       }
3899     }
3900   }
3901 }
3902 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3903   \tl_put_right:Nx \l_tmpc_tl {
3904     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3905   }
3906 }
3907 \tl_put_right:Nx \l_tmpb_tl {
3908   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3909 }
3910 }
3911 }
3912 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3913 \tl_put_left:Nx \l_tmpa_tl {
3914   \prop_set_from_keyval:cn {
3915     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
3916   }{
3917     \prop_to_keyval:N \l_stex_current_copymodule_prop
3918   }
3919 }
3920 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3921 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3922 \exp_args:Nx \stex_do_up_to_module:n {
3923   \exp_args:No \exp_not:n \l_tmpa_tl
3924 }
3925 \l_tmpb_tl
3926 \stex_if_smsmode:F {
3927   \end{stex_annotate_env}
3928 }
3929 }
3930
3931 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3932   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3933   \stex_deactivate_macro:Nn \symdecl {module~environments}
3934   \stex_deactivate_macro:Nn \symdef {module~environments}
3935   \stex_deactivate_macro:Nn \notation {module~environments}
3936   \stex_reactivate_macro:N \assign
3937   \stex_reactivate_macro:N \renamedekl
3938   \stex_reactivate_macro:N \donotcopy
3939   \stex_smsmode_do:
3940 }{
3941   \stex_copymodule_end:n {}
3942 }
3943
3944 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3945   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3946   \stex_deactivate_macro:Nn \symdecl {module~environments}
3947   \stex_deactivate_macro:Nn \symdef {module~environments}
3948   \stex_deactivate_macro:Nn \notation {module~environments}
3949   \stex_reactivate_macro:N \assign
3950   \stex_reactivate_macro:N \renamedekl
3951   \stex_reactivate_macro:N \donotcopy
3952   \stex_smsmode_do:
3953 }{
3954   \stex_copymodule_end:n {
3955     \tl_if_exist:cF {
3956       l__stex_copymodule_copymodule_##1?##2_def_tl
3957     }{
3958       \str_if_eq:eeF {
3959         \prop_item:cn{
3960           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
3961         }{ true }{
3962           \msg_error:nnxx{stex}{error/interpretmodule/nodéfiniens}{
3963             ##1?##2
3964           }{\l_stex_current_copymodule_name_str}
3965         }
3966       }
3967     }
3968   }
3969
3970 \NewDocumentCommand \donotcopy { 0{} m}{
3971   \stex_import_module_uri:nn { #1 } { #2 }
3972   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3973   \seq_map_inline:Nn \l_stex_collect_imports_seq {
3974     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }

```



```

3975 \seq_map_inline:cn {c_stex_module_##1_constants}{
3976 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
3977 \bool_lazy_any_p:nT {
3978   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
3979   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3980   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
3981 }{
3982   % TODO throw error
3983 }
3984 }
3985 }
3986
3987 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
3988 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
3989 \prop_put:Nnx \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
3990 }
3991
3992 \NewDocumentCommand \assign { m m }{
3993   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
3994   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
3995   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
3996 }
3997
3998 \keys_define:nn { stex / renamedec1 } {
3999   name .str_set_x:N = \l_stex_renamedec1_name_str
4000 }
4001 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4002   \str_clear:N \l_stex_renamedec1_name_str
4003   \keys_set:nn { stex / renamedec1 } { #1 }
4004 }
4005
4006 \NewDocumentCommand \renamedec1 { 0{} m m }{
4007   \__stex_copymodule_renamedec1_args:n { #1 }
4008   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4009   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4010   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4011   \str_if_empty:NTF \l_stex_renamedec1_name_str {
4012     \tl_set:cx { #3 }{\stex_invoke_symbol:n {
4013       \l_stex_get_symbol_uri_str
4014     } }
4015   } {
4016     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4017     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4018     \prop_set_eq:cc {l_stex_symdec1_
4019       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4020     _prop
4021     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
4022     \seq_set_eq:cc {l_stex_symdec1_
4023       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4024     _notations
4025     }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
4026     \prop_put:cnx {l_stex_symdec1_
4027       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4028     _prop

```

```

4029     }{ name }{ \l_stex_renamedecl_name_str }
4030     \prop_put:cnx {l_stex_symdecl_
4031       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4032       _prop
4033     }{ module }{ \l_stex_current_module_str }
4034     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4035       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4036     }
4037     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4038       \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4039     } }
4040   }
4041 }
4042
4043 \stex_deactivate_macro:Nn \assign {copymodules}
4044 \stex_deactivate_macro:Nn \renamedecl {copymodules}
4045 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4046
4047
4048 \seq_new:N \l_stex_implicit_morphisms_seq
4049 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4050   \stex_import_module_uri:nn { #1 } { #2 }
4051   \stex_debug:nn{implicits}{
4052     Implicit~morphism:~
4053     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4054   }
4055   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4056     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4057   }{
4058     \msg_error:nnn{stex}{error/conflictingmodules}{
4059       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4060     }
4061   }
4062
4063   % TODO
4064
4065
4066   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4067     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4068   }
4069 }
4070 }
4071

```

31.2 The feature environment

structural@feature

```

4072 <@@=stex_features>
4073
4074 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4075   \stex_if_in_module:F {
4076     \msg_set:nnn{stex}{error/nomodule}{
4077       Structural~Feature~has~to~occur~in~a~module:\\

```

```

4078     Feature~#2~of~type~#1\\
4079     In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4080   }
4081   \msg_error:nn{stex}{error/nomodule}
4082 }
4083
4084 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4085
4086 \stex_if_smsmode:F {
4087   \begin{stex_annotate_env}{ feature:#1 }{}
4088   \stex_annotate_invisible:nnn{header}{}{ #3 }
4089 }
4090 }{
4091   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4092   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4093   \stex_debug:nn{features}{
4094     Feature: \l_stex_last_feature_str
4095   }
4096   \stex_if_smsmode:F {
4097     \end{stex_annotate_env}
4098   }
4099 }

```

31.3 Structure

structure

```

4100 <@@=stex_structures>
4101 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4102   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4103     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4104   }
4105   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4106   {#1}{#2}
4107 }
4108
4109 \keys_define:nn { stex / features / structure } {
4110   name .str_set_x:N = \l__stex_structures_name_str ,
4111 }
4112
4113 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4114   \str_clear:N \l__stex_structures_name_str
4115   \keys_set:nn { stex / features / structure } { #1 }
4116 }
4117
4118 \NewDocumentEnvironment{mathstructure}{m O{}}{
4119   \__stex_structures_structure_args:n { #2 }
4120   \str_if_empty:NT \l__stex_structures_name_str {
4121     \str_set:Nx \l__stex_structures_name_str { #1 }
4122   }
4123   \exp_args:Nx \stex_symdecl_do:nn {
4124     name = \l__stex_structures_name_str ,
4125     type = \metacollection ,
4126     def = {\STEXsymbol{module-type}}{

```

```

4127     \stex_term_math_oms:nnnn {
4128         \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
4129         { ns } \l_stex_module_ns_str ?
4130         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4131         { name } / \l__stex_structures_name_str - structure
4132     }{}{}{}
4133 }
4134 }{ #1 }
4135 \exp_args:Nnnx
4136 \begin{structural_feature_module}{ structure }
4137 { \l__stex_structures_name_str }{}
4138 \stex_smsmode_do:
4139 }{
4140     \end{structural_feature_module}
4141     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4142     \seq_clear:N \l_tmpa_seq
4143     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4144         \seq_map_inline:cn{c_stex_module_##1_constants}{
4145             \seq_put_right:Nn \l_tmpa_seq { ##1 ? #####1 }
4146         }
4147     }
4148     \exp_args:Nnno
4149     \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4150     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4151     \stex_add_structure_to_current_module:nn
4152         \l_stex_structures_name_str
4153         \l_stex_last_feature_str
4154     \exp_args:Nx
4155     \stex_add_to_current_module:n {
4156         \tl_set:cn { #1 }{
4157             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4158         }
4159     }
4160     \exp_args:Nx
4161     \stex_do_up_to_module:n {
4162         \tl_set:cn { #1 }{
4163             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4164         }
4165     }
4166 }
4167 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4168
4169 \cs_new:Nn \stex_invoke_structure:nn {
4170     \stex_invoke_symbol:n { #1?#2 }
4171 }
4172
4173 \cs_new_protected:Nn \stex_get_structure:n {
4174     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4175         \tl_set:Nn \l_tmpa_tl { #1 }
4176         \__stex_structures_get_from_cs:
4177     }{
4178         \cs_if_exist:cTF { #1 }{
4179             \cs_set_eq:Nc \l_tmpa_cs { #1 }
4180             \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }

```

```

4181 \str_if_empty:NTF \l_tmpa_str {
4182 \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4183 \__stex_structures_get_from_cs:
4184 }{
4185 \__stex_structures_get_from_string:n { #1 }
4186 }
4187 }{
4188 \__stex_structures_get_from_string:n { #1 }
4189 }
4190 }{
4191 \__stex_structures_get_from_string:n { #1 }
4192 }
4193 }
4194 }
4195
4196 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4197 \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4198 { \tl_tail:N \l_tmpa_tl }
4199 \str_set:Nx \l_tmpa_str {
4200 \exp_after:wN \use_i:nn \l_tmpa_tl
4201 }
4202 \str_set:Nx \l_tmpb_str {
4203 \exp_after:wN \use_ii:nn \l_tmpa_tl
4204 }
4205 \str_set:Nx \l_stex_get_structure_str {
4206 \l_tmpa_str ? \l_tmpb_str
4207 }
4208 \str_set:Nx \l_stex_get_structure_module_str {
4209 \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4210 }
4211 }
4212
4213 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4214 \tl_set:Nn \l_tmpa_tl {
4215 \msg_error:nnn{stex}{error/unknownstructure}{#1}
4216 }
4217 \str_set:Nn \l_tmpa_str { #1 }
4218 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4219
4220 \seq_map_inline:Nn \l_stex_all_modules_seq {
4221 \prop_if_exist:cT {c_stex_module_##1_structures} {
4222 \prop_map_inline:cn {c_stex_module_##1_structures} {
4223 \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4224 \prop_map_break:n{ \seq_map_break:n{
4225 \tl_set:Nn \l_tmpa_tl {
4226 \str_set:Nn \l_stex_get_structure_str {##1?####1}
4227 \str_set:Nn \l_stex_get_structure_module_str {####2}
4228 }
4229 }}
4230 }
4231 }
4232 }
4233 }
4234 \l_tmpa_tl

```

4235 }

\instantiate

4236

4237 \keys_define:nn { stex / instantiate } {

4238 name .str_set_x:N = \l__stex_structures_name_str

4239 }

4240 \cs_new_protected:Nn __stex_structures_instantiate_args:n {

4241 \str_clear:N \l__stex_structures_name_str

4242 \keys_set:nn { stex / instantiate } { #1 }

4243 }

4244

4245 \NewDocumentCommand \instantiate {m O{} m m m}{

4246 \begin{group}

4247 \stex_get_structure:n {#4}

4248 __stex_structures_instantiate_args:n { #2 }

4249 \str_if_empty:NT \l__stex_structures_name_str {

4250 \str_set:Nn \l__stex_structures_name_str { #1 }

4251 }

4252 \seq_clear:N \l__stex_structures_fields_seq

4253 \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str

4254 \seq_map_inline:Nn \l_stex_collect_imports_seq {

4255 \seq_map_inline:cn {c_stex_module_##1_constants}{

4256 \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }

4257 }

4258 }

4259 \seq_set_split:Nnn \l_tmpa_seq , {#3}

4260 \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str

4261 \prop_clear:N \l_tmpa_prop

4262 \seq_map_inline:Nn \l_tmpa_seq {

4263 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }

4264 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {

4265 \msg_error:nnn{stex}{error/keyval}{##1}

4266 }

4267 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_fields_seq

4268 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str

4269 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str

4270 \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}

4271 \exp_args:Nxx \str_if_eq:nnF

4272 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}

4273 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{

4274 \msg_error:nnxxx{stex}{error/incompatible}

4275 {\l__stex_structures_dom_str

4276 {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}

4277 {\l_stex_get_symbol_uri_str

4278 {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}}

4279 }

4280 \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str

4281 }

4282 \seq_if_empty:NF \l__stex_structures_fields_seq {

4283 \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_seq}

4284 }

4285 \exp_args:Nx

4286 \stex_add_to_current_module:n {

```

4287     \prop_set_from_keyval:cn {l_stex_instance~\l_stex_current_module_str?\l__stex_structur
4288         domain = \l_stex_get_structure_module_str ,
4289         \prop_to_keyval:N \l_tmpa_prop
4290     }
4291     \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4292 }
4293 \exp_args:Nx
4294 \stex_do_up_to_module:n {
4295     \prop_set_from_keyval:cn {l_stex_instance~\l_stex_current_module_str?\l__stex_structur
4296         domain = \l_stex_get_structure_module_str ,
4297         \prop_to_keyval:N \l_tmpa_prop
4298     }
4299     \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4300 }
4301 \stex_debug:nn{instantiate}{
4302     Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4303     \prop_to_keyval:N \l_tmpa_prop
4304 }
4305 \exp_args:Nxx \stex_symdecl_do:nn {
4306     type={\STEXsymbol{module-type}}{
4307         \_stex_term_math_oms:nnnn {
4308             \l_stex_get_structure_module_str
4309         }{}{0}{}
4310     }}
4311     }{\l__stex_structures_name_str}
4312     \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4313 \endgroup
4314 \stex_smsmode_do:\ignorespacesandpars
4315 }
4316 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4317
4318 \cs_new_protected:Nn \stex_symbol_or_var:n {
4319     \cs_if_exist:cTF{#1}{
4320         \cs_set_eq:Nc \l_tmpa_tl { #1 }
4321         \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4322         \str_if_empty:NTF \l_tmpa_str {
4323             \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4324                 \stex_invoke_variable:n {
4325                     \bool_set_true:N \l_stex_symbol_or_var_bool
4326                     \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4327                     \str_set:Nx \l_stex_get_symbol_uri_str {
4328                         \exp_after:wN \use:n \l_tmpa_tl
4329                     }
4330                 }{
4331                     \bool_set_false:N \l_stex_symbol_or_var_bool
4332                     \stex_get_symbol:n{#1}
4333                 }
4334             }{
4335                 \__stex_structures_symbolorvar_from_string:n{ #1 }
4336             }
4337         }{
4338             \__stex_structures_symbolorvar_from_string:n{ #1 }
4339         }
4340     }

```

```

4341
4342 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4343   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4344     \bool_set_true:N \l_stex_symbol_or_var_bool
4345     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4346   }{
4347     \bool_set_false:N \l_stex_symbol_or_var_bool
4348     \stex_get_symbol:n{#1}
4349   }
4350 }
4351
4352
4353 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4354   \begin_group
4355     \stex_get_structure:n {#4}
4356     \__stex_structures_instantiate_args:n { #2 }
4357     \str_if_empty:NT \l__stex_structures_name_str {
4358       \str_set:Nn \l__stex_structures_name_str { #1 }
4359     }
4360     \seq_clear:N \l__stex_structures_fields_seq
4361     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4362     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4363       \seq_map_inline:cn {c_stex_module_##1_constants}{
4364         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4365       }
4366     }
4367     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4368     \prop_clear:N \l_tmpa_prop
4369     \tl_if_empty:nF {#3} {
4370       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4371       \seq_map_inline:Nn \l_tmpa_seq {
4372         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4373         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4374           \msg_error:nnn{stex}{error/keyval}{##1}
4375         }
4376         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4377         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4378         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4379         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4380         \bool_if:NTF \l_stex_symbol_or_var_bool {
4381           \exp_args:Nxx \str_if_eq:nnF
4382             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4383             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4384             \msg_error:nnxxx{stex}{error/incompatible}
4385             {\l__stex_structures_dom_str}
4386             {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4387             {\l_stex_get_symbol_uri_str}
4388             {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4389           }
4390         \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4391       }{
4392         \exp_args:Nxx \str_if_eq:nnF
4393           {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4394           {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{

```



```

4395         \msg_error:nnxxxx{stex}{error/incompatible}
4396         {\l__stex_structures_dom_str}
4397         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str_prop}{args}}
4398         {\l_stex_get_symbol_uri_str}
4399         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str_prop}{args}}
4400     }
4401     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4402 }
4403 }
4404 }
4405 \tl_gclear:N \g__stex_structures_aftergroup_tl
4406 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4407     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl
4408     \stex_find_notation:nn{##1}{}}
4409     \cs_gset_eq:cc{g__stex_structures_tmpa\l_tmpa_str_cs}
4410     {stex_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4411     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}{
4412     \cs_gset_eq:cc {g__stex_structures_tmpa_op\l_tmpa_str_cs}
4413     {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str_cs}
4414 }
4415 }
4416 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4417     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str_prop}{
4418         name = \l_tmpa_str ,
4419         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4420         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4421         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4422     }
4423     \cs_set_eq:cc {stex_var_notation\l_tmpa_str_cs}
4424     {g__stex_structures_tmpa\l_tmpa_str_cs}
4425     \cs_set_eq:cc {stex_var_op_notation\l_tmpa_str_cs}
4426     {g__stex_structures_tmpa_op\l_tmpa_str_cs}
4427 }
4428 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invo
4429 }
4430 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4431     \prop_set_from_keyval:cn {l_stex_varinstance\l__stex_structures_name_str_prop }{
4432         domain = \l_stex_get_structure_module_str ,
4433         \prop_to_keyval:N \l_tmpa_prop
4434     }
4435     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4436     \tl_set:cn {l_stex_varinstance\l__stex_structures_name_str_op_tl}{
4437         \exp_args:Nnx \exp_not:N \use:nn {
4438             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name
4439             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4440                 \exp_not:n{
4441                     \_varcomp{#5}
4442                 }
4443             }
4444         }{
4445             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4446         }
4447     }
4448 }

```

```

4449 \aftergroup\g__stex_structures_aftergroup_tl
4450 \endgroup
4451 \stex_smsmode_do:\ignorespacesandpars
4452 }
4453
4454 \cs_new_protected:Nn \stex_invoke_instance:n {
4455   \peek_charcode_remove:NTF ! {
4456     \stex_invoke_symbol:n{#1}
4457   }{
4458     \stex_invoke_instance:nn {#1}
4459   }
4460 }
4461
4462
4463 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4464   \peek_charcode_remove:NTF ! {
4465     \use:c{l_stex_varinstance_#1_op_tl}
4466   }{
4467     \stex_invoke_varinstance:nn {#1}
4468   }
4469 }
4470
4471 \cs_new_protected:Nn \stex_invoke_instance:nn {
4472   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4473     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4474   }{
4475     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4476     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4477       \prop_to_keyval:N \l_tmpa_prop
4478     }
4479   }
4480 }
4481
4482 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4483   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4484     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4485     \l_tmpa_tl
4486   }{
4487     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4488   }
4489 }

```

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

\stex_invoke_structure:nnn

```

4490 % #1: URI of the instance
4491 % #2: URI of the instantiated module
4492 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4493   \tl_if_empty:nTF{ #3 }{
4494     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4495       c_stex_feature_ #2 _prop
4496     }
4497     \tl_clear:N \l_tmpa_tl
4498     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq

```

```

4499 \seq_map_inline:Nn \l_tmpa_seq {
4500   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4501   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4502   \cs_if_exist:cT {
4503     stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4504   }{
4505     \tl_if_empty:NF \l_tmpa_tl {
4506       \tl_put_right:Nn \l_tmpa_tl {,}
4507     }
4508     \tl_put_right:Nx \l_tmpa_tl {
4509       \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4510     }
4511   }
4512 }
4513 \exp_args:No \mathstrut \l_tmpa_tl
4514 }{
4515   \stex_invoke_symbol:n{#1/#3}
4516 }
4517 }

```

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

```

4518 </package>

```

Chapter 32

STEX -Statements Implementation

```
4519 <*package>
4520
4521 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4522
4523 <@@=stex_statements>
    Warnings and error messages
4524

\titleemph
4525 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
4526 \keys_define:nn {stex / definiendum }{
4527   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4528   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4529   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4530   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4531 }
4532 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4533   \str_clear:N \l__stex_statements_definiendum_root_str
4534   \tl_clear:N \l__stex_statements_definiendum_post_tl
4535   \str_clear:N \l__stex_statements_definiendum_gfa_str
4536   \keys_set:nn { stex / definiendum }{ #1 }
4537 }
4538 \NewDocumentCommand \definiendum { O{} m m } {
4539   \__stex_statements_definiendum_args:n { #1 }
4540   \stex_get_symbol:n { #2 }
4541   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4542   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4543     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4544     \tl_set:Nn \l_tmpa_tl { #3 }
4545   } {
4546     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4547     \tl_set:Nn \l_tmpa_tl {
4548       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4549     }
4550   }
4551 } {
4552   \tl_set:Nn \l_tmpa_tl { #3 }
4553 }
4554
4555 % TODO root
4556 \rustex_if:TF {
4557   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4558 } {
4559   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4560 }
4561 }
4562 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4563
4564 \NewDocumentCommand \definame { 0{ } m } {
4565   \__stex_statements_definiendum_args:n { #1 }
4566   % TODO: root
4567   \stex_get_symbol:n { #2 }
4568   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4569   \str_set:Nx \l_tmpa_str {
4570     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4571   }
4572   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4573   \rustex_if:TF {
4574     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4575       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4576     }
4577   } {
4578     \exp_args:Nnx \defemph@uri {
4579       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4580     } { \l_stex_get_symbol_uri_str }
4581   }
4582 }
4583 \stex_deactivate_macro:Nn \definame {definition~environments}
4584
4585 \NewDocumentCommand \Definame { 0{ } m } {
4586   \__stex_statements_definiendum_args:n { #1 }
4587   \stex_get_symbol:n { #2 }
4588   \str_set:Nx \l_tmpa_str {
4589     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4590   }
4591   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4592   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4593   \rustex_if:TF {

```

```

4594 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4595 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4596 }
4597 } {
4598 \exp_args:Nnx \defemph@uri {
4599 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4600 } { \l_stex_get_symbol_uri_str }
4601 }
4602 }
4603 \stex_deactivate_macro:Nn \Definame {definition~environments}
4604
4605 \NewDocumentCommand \premise { m }{
4606 \stex_annotate:nnn{ premise }{}{ #1 }
4607 }
4608 \NewDocumentCommand \conclusion { m }{
4609 \stex_annotate:nnn{ conclusion }{}{ #1 }
4610 }
4611 \NewDocumentCommand \definiens { O{} m }{
4612 \str_clear:N \l_stex_get_symbol_uri_str
4613 \tl_if_empty:nF {#1} {
4614 \stex_get_symbol:n { #1 }
4615 }
4616 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4617 }
4618
4619 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4620 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4621 \stex_deactivate_macro:Nn \definiens {definition~environments}
4622

```

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

sdefinition

```

4623
4624 \keys_define:nn {stex / sdefinition }{
4625 type .str_set_x:N = \sdefinitiontype,
4626 id .str_set_x:N = \sdefinitionid,
4627 name .str_set_x:N = \sdefinitionname,
4628 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4629 title .tl_set:N = \sdefinitiontitle
4630 }
4631 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4632 \str_clear:N \sdefinitiontype
4633 \str_clear:N \sdefinitionid
4634 \str_clear:N \sdefinitionname
4635 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4636 \tl_clear:N \sdefinitiontitle
4637 \keys_set:nn { stex / sdefinition }{ #1 }
4638 }
4639
4640 \NewDocumentEnvironment{sdefinition}{O{} }{
4641 \__stex_statements_sdefinition_args:n{ #1 }
4642 \stex_reactivate_macro:N \definiendum
4643 \stex_reactivate_macro:N \definame

```

```

4644 \stex_reactivate_macro:N \Definame
4645 \stex_reactivate_macro:N \premise
4646 \stex_reactivate_macro:N \definiens
4647 \stex_if_smsmode:F{
4648   \seq_clear:N \l_tmpa_seq
4649   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4650     \tl_if_empty:nF{ ##1 }{
4651       \stex_get_symbol:n { ##1 }
4652       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4653         \l_stex_get_symbol_uri_str
4654       }
4655     }
4656   }
4657   \exp_args:Nnnx
4658   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4659   \str_if_empty:NF \sdefinitiontype {
4660     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{ }
4661   }
4662   \clist_set:N \l_tmpa_clist \sdefinitiontype
4663   \tl_clear:N \l_tmpa_tl
4664   \clist_map_inline:Nn \l_tmpa_clist {
4665     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4666       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4667     }
4668   }
4669   \tl_if_empty:NTF \l_tmpa_tl {
4670     \__stex_statements_sdefinition_start:
4671   }{
4672     \l_tmpa_tl
4673   }
4674 }
4675 \stex_ref_new_doc_target:n \sdefinitionid
4676 \stex_smsmode_do:
4677 ){
4678   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4679   \stex_if_smsmode:F {
4680     \clist_set:N \l_tmpa_clist \sdefinitiontype
4681     \tl_clear:N \l_tmpa_tl
4682     \clist_map_inline:Nn \l_tmpa_clist {
4683       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4684         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4685       }
4686     }
4687     \tl_if_empty:NTF \l_tmpa_tl {
4688       \__stex_statements_sdefinition_end:
4689     }{
4690       \l_tmpa_tl
4691     }
4692     \end{stex_annotate_env}
4693   }
4694 }

```

\stexpatchdefinition

```

4695 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {

```

```

4696 \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4697   ~(\sdefinitiontitle)
4698 }~}
4699 }
4700 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4701
4702 \newcommand\stexpatchdefinition[3][] {
4703   \str_set:Nx \l_tmpa_str{ #1 }
4704   \str_if_empty:NTF \l_tmpa_str {
4705     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4706     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4707   }{
4708     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4709     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4710   }
4711 }

```

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

\inlinedef inline:

```

4712 \keys_define:nn {stex / inlinedef }{
4713   type      .str_set_x:N = \sdefinitiontype,
4714   id        .str_set_x:N = \sdefinitionid,
4715   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4716   name      .str_set_x:N = \sdefinitionname
4717 }
4718 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4719   \str_clear:N \sdefinitiontype
4720   \str_clear:N \sdefinitionid
4721   \str_clear:N \sdefinitionname
4722   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4723   \keys_set:nn { stex / inlinedef }{ #1 }
4724 }
4725 \NewDocumentCommand \inlinedef { 0{} m } {
4726   \begingroup
4727   \__stex_statements_inlinedef_args:n{ #1 }
4728   \stex_reactivate_macro:N \definiendum
4729   \stex_reactivate_macro:N \definame
4730   \stex_reactivate_macro:N \Definame
4731   \stex_reactivate_macro:N \premise
4732   \stex_reactivate_macro:N \definiens
4733   \stex_ref_new_doc_target:n \sdefinitionid
4734   \stex_if_smsmode:TF{
4735     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4736   }{
4737     \seq_clear:N \l_tmpa_seq
4738     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4739       \tl_if_empty:nF{ ##1 }{
4740         \stex_get_symbol:n { ##1 }
4741         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4742           \l_stex_get_symbol_uri_str
4743         }
4744       }
4745     }

```



```

4746 \exp_args:Nnx
4747 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4748   \str_if_empty:NF \sdefinitiontype {
4749     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4750   }
4751   #2
4752   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}}{\sdefinitionname} }
4753 }
4754 }
4755 \endgroup
4756 \stex_smsmode_do:
4757 }

```

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

32.2 Assertions

sassertion

```

4758
4759 \keys_define:nn {stex / sassertion }{
4760   type      .str_set_x:N = \sassertiontype,
4761   id        .str_set_x:N = \sassertionid,
4762   title     .tl_set:N    = \sassertiontitle ,
4763   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4764   name      .str_set_x:N = \sassertionname
4765 }
4766 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4767   \str_clear:N \sassertiontype
4768   \str_clear:N \sassertionid
4769   \str_clear:N \sassertionname
4770   \clist_clear:N \l__stex_statements_sassertion_for_clist
4771   \tl_clear:N \sassertiontitle
4772   \keys_set:nn { stex / sassertion }{ #1 }
4773 }
4774
4775 %\tl_new:N \g__stex_statements_aftergroup_tl
4776
4777 \NewDocumentEnvironment{sassertion}{0{}}{
4778   \__stex_statements_sassertion_args:n{ #1 }
4779   \stex_reactivate_macro:N \premise
4780   \stex_reactivate_macro:N \conclusion
4781   \stex_if_smsmode:F {
4782     \seq_clear:N \l_tmpa_seq
4783     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4784       \tl_if_empty:nF{ ##1 }{
4785         \stex_get_symbol:n { ##1 }
4786         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4787           \l_stex_get_symbol_uri_str
4788         }
4789       }
4790     }
4791   }
4792   \exp_args:Nnnx
4793   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}

```

```

4793 \str_if_empty:NF \sassertiontype {
4794   \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4795 }
4796 \clist_set:No \l_tmpa_clist \sassertiontype
4797 \tl_clear:N \l_tmpa_tl
4798 \clist_map_inline:Nn \l_tmpa_clist {
4799   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4800     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4801   }
4802 }
4803 \tl_if_empty:NTF \l_tmpa_tl {
4804   \__stex_statements_sassertion_start:
4805 }{
4806   \l_tmpa_tl
4807 }
4808 }
4809 \str_if_empty:NTF \sassertionid {
4810   \str_if_empty:NF \sassertionname {
4811     \stex_ref_new_doc_target:n {}
4812   }
4813 } {
4814   \stex_ref_new_doc_target:n \sassertionid
4815 }
4816 \stex_smsmode_do:
4817 }{
4818   \str_if_empty:NF \sassertionname {
4819     \stex_symdecl_do:nn{ }\sassertionname}
4820     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4821   }
4822   \stex_if_smsmode:F {
4823     \clist_set:No \l_tmpa_clist \sassertiontype
4824     \tl_clear:N \l_tmpa_tl
4825     \clist_map_inline:Nn \l_tmpa_clist {
4826       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4827         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4828       }
4829     }
4830     \tl_if_empty:NTF \l_tmpa_tl {
4831       \__stex_statements_sassertion_end:
4832     }{
4833       \l_tmpa_tl
4834     }
4835     \end{stex_annotate_env}
4836   }
4837 }

```

\stexpatchassertion

```

4838
4839 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4840   \par\noindent\titleemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4841     (\sassertiontitle)
4842   }~}
4843 }
4844 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}

```

```

4845
4846 \newcommand\stexpatchassertion[3] [] {
4847   \str_set:Nx \l_tmpa_str{ #1 }
4848   \str_if_empty:NTF \l_tmpa_str {
4849     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4850     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4851   }{
4852     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4853     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4854   }
4855 }

```

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

\inlineass inline:

```

4856 \keys_define:nn {stex / inlineass }{
4857   type      .str_set_x:N = \sassertiontype,
4858   id        .str_set_x:N = \sassertionid,
4859   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4860   name      .str_set_x:N = \sassertionname
4861 }
4862 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4863   \str_clear:N \sassertiontype
4864   \str_clear:N \sassertionid
4865   \str_clear:N \sassertionname
4866   \clist_clear:N \l__stex_statements_sassertion_for_clist
4867   \keys_set:nn { stex / inlineass }{ #1 }
4868 }
4869 \NewDocumentCommand \inlineass { 0{} m } {
4870   \begingroup
4871   \stex_reactivate_macro:N \premise
4872   \stex_reactivate_macro:N \conclusion
4873   \__stex_statements_inlineass_args:n{ #1 }
4874   \str_if_empty:NTF \sassertionid {
4875     \str_if_empty:NF \sassertionname {
4876       \stex_ref_new_doc_target:n {}
4877     }
4878   } {
4879     \stex_ref_new_doc_target:n \sassertionid
4880   }
4881
4882   \stex_if_smsmode:TF{
4883     \str_if_empty:NF \sassertionname {
4884       \stex_symdecl_do:nn{}{\sassertionname}
4885       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4886     }
4887   }{
4888     \seq_clear:N \l_tmpa_seq
4889     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4890       \tl_if_empty:NF{ ##1 }{
4891         \stex_get_symbol:n { ##1 }
4892         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4893           \l_stex_get_symbol_uri_str
4894         }

```

```

4895     }
4896   }
4897   \exp_args:Nnx
4898   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4899     \str_if_empty:NF \sassertiontype {
4900       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4901   }
4902   #2
4903   \str_if_empty:NF \sassertionname {
4904     \stex_symdecl_do:nn{}{\sassertionname}
4905     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4906   }
4907 }
4908 }
4909 \endgroup
4910 \stex_smsmode_do:
4911 }

```

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

32.3 Examples

`sexample`

```

4912
4913 \keys_define:nn {stex / sexample }{
4914   type      .str_set_x:N = \exampletype,
4915   id        .str_set_x:N = \sexampleid,
4916   title     .tl_set:N     = \sexampletile,
4917   name      .str_set_x:N = \sexamplename ,
4918   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4919 }
4920 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4921   \str_clear:N \sexampletype
4922   \str_clear:N \sexampleid
4923   \str_clear:N \sexamplename
4924   \tl_clear:N \sexampletile
4925   \clist_clear:N \l__stex_statements_sexample_for_clist
4926   \keys_set:nn { stex / sexample }{ #1 }
4927 }
4928
4929 \NewDocumentEnvironment{sexample}{0{}}{
4930   \__stex_statements_sexample_args:n{ #1 }
4931   \stex_reactivate_macro:N \premise
4932   \stex_reactivate_macro:N \conclusion
4933   \stex_if_smsmode:F {
4934     \seq_clear:N \l_tmpa_seq
4935     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
4936       \tl_if_empty:nF{ ##1 }{
4937         \stex_get_symbol:n { ##1 }
4938         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4939           \l_stex_get_symbol_uri_str
4940         }
4941       }

```

```

4942 }
4943 \exp_args:Nnnx
4944 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
4945 \str_if_empty:NF \sexamplotype {
4946   \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
4947 }
4948 \clist_set:No \l_tmpa_clist \sexamplotype
4949 \tl_clear:N \l_tmpa_tl
4950 \clist_map_inline:Nn \l_tmpa_clist {
4951   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
4952     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
4953   }
4954 }
4955 \tl_if_empty:NTF \l_tmpa_tl {
4956   \__stex_statements_sexample_start:
4957 }{
4958   \l_tmpa_tl
4959 }
4960 }
4961 \str_if_empty:NF \sexampleid {
4962   \stex_ref_new_doc_target:n \sexampleid
4963 }
4964 \stex_smsmode_do:
4965 }{
4966   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
4967   \stex_if_smsmode:F {
4968     \clist_set:No \l_tmpa_clist \sexamplotype
4969     \tl_clear:N \l_tmpa_tl
4970     \clist_map_inline:Nn \l_tmpa_clist {
4971       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
4972         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
4973       }
4974     }
4975     \tl_if_empty:NTF \l_tmpa_tl {
4976       \__stex_statements_sexample_end:
4977     }{
4978       \l_tmpa_tl
4979     }
4980     \end{stex_annotate_env}
4981   }
4982 }

```

\stexpatchexample

```

4983
4984 \cs_new_protected:Nn \__stex_statements_sexample_start: {
4985   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplename {
4986     (\sexamplename)
4987   }~}
4988 }
4989 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
4990
4991 \newcommand\stexpatchexample[3] [] {
4992   \str_set:Nx \l_tmpa_str{ #1 }
4993   \str_if_empty:NTF \l_tmpa_str {

```

```

4994     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
4995     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
4996   }{
4997     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
4998     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
4999   }
5000 }

```

\inlineex inline:

```

5044 \stex_smsmode_do:
5045 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5046 \keys_define:nn { stex / sparagraph} {
5047   id      .str_set_x:N = \sparagraphid ,
5048   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5049   type    .str_set_x:N = \sparagraphtype ,
5050   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5051   from    .tl_set:N    = \sparagraphfrom ,
5052   to      .tl_set:N    = \sparagraphto ,
5053   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5054   name    .str_set:N   = \sparagraphname
5055 }
5056
5057 \cs_new_protected:Nn \stex_sparagraph_args:n {
5058   \tl_clear:N \l_stex_sparagraph_title_tl
5059   \tl_clear:N \sparagraphfrom
5060   \tl_clear:N \sparagraphto
5061   \tl_clear:N \l_stex_sparagraph_start_tl
5062   \str_clear:N \sparagraphid
5063   \str_clear:N \sparagraphtype
5064   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5065   \str_clear:N \sparagraphname
5066   \keys_set:nn { stex / sparagraph }{ #1 }
5067 }
5068 \newif\if@in@omtext\@in@omtextfalse
5069
5070 \NewDocumentEnvironment {sparagraph} { 0{} } {
5071   \stex_sparagraph_args:n { #1 }
5072   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5073     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5074   }{
5075     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5076   }
5077   \@in@omtexttrue
5078   \stex_if_smsmode:F {
5079     \seq_clear:N \l_tmpa_seq
5080     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5081       \tl_if_empty:nF{ ##1 }{
5082         \stex_get_symbol:n { ##1 }
5083         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5084           \l_stex_get_symbol_uri_str
5085         }
5086       }
5087     }
5088     \exp_args:Nnnx
5089     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5090     \str_if_empty:NF \sparagraphtype {

```

```

5091     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5092   }
5093   \str_if_empty:NF \sparagraphfrom {
5094     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5095   }
5096   \str_if_empty:NF \sparagraphto {
5097     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5098   }
5099   \clist_set:No \l_tmpa_clist \sparagraphtype
5100   \tl_clear:N \l_tmpa_tl
5101   \clist_map_inline:Nn \sparagraphtype {
5102     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5103       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5104     }
5105   }
5106   \tl_if_empty:NTF \l_tmpa_tl {
5107     \__stex_statements_sparagraph_start:
5108   }{
5109     \l_tmpa_tl
5110   }
5111 }
5112 \clist_set:No \l_tmpa_clist \sparagraphtype
5113 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5114 {
5115   \stex_reactivate_macro:N \definiendum
5116   \stex_reactivate_macro:N \definame
5117   \stex_reactivate_macro:N \Definame
5118   \stex_reactivate_macro:N \premise
5119   \stex_reactivate_macro:N \definiens
5120 }
5121 \str_if_empty:NTF \sparagraphid {
5122   \str_if_empty:NTF \sparagraphname {
5123     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5124       \stex_ref_new_doc_target:n {}
5125     }
5126   } {
5127     \stex_ref_new_doc_target:n {}
5128   }
5129 } {
5130   \stex_ref_new_doc_target:n \sparagraphid
5131 }
5132 \exp_args:NNx
5133 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5134   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5135     \tl_if_empty:nF{ ##1 }{
5136       \stex_get_symbol:n { ##1 }
5137       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5138     }
5139   }
5140 }
5141 \stex_smsmode_do:
5142 \ignorespacesandpars
5143 }{
5144   \str_if_empty:NF \sparagraphname {

```



```

5145 \stex_symdecl_do:nn{}{\sparagraphname}
5146 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5147 }
5148 \stex_if_smsmode:F {
5149 \clist_set:N \l_tmpa_clist \sparagraphtype
5150 \tl_clear:N \l_tmpa_tl
5151 \clist_map_inline:Nn \l_tmpa_clist {
5152 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5153 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5154 }
5155 }
5156 \tl_if_empty:NTF \l_tmpa_tl {
5157 \__stex_statements_sparagraph_end:
5158 }{
5159 \l_tmpa_tl
5160 }
5161 \end{stex_annotate_env}
5162 }
5163 }

```

\stexpatchparagraph

```

5164
5165 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5166 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5167 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5168 \titleemph{\l_stex_sparagraph_title_tl}:~
5169 }
5170 }{
5171 \titleemph{\l_stex_sparagraph_start_tl}~
5172 }
5173 }
5174 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5175
5176 \newcommand\stexpatchparagraph[3] [] {
5177 \str_set:Nx \l_tmpa_str{ #1 }
5178 \str_if_empty:NTF \l_tmpa_str {
5179 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5180 \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5181 }{
5182 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5183 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5184 }
5185 }
5186
5187 \keys_define:nn { stex / inlinepara} {
5188 id .str_set_x:N = \sparagraphid ,
5189 type .str_set_x:N = \sparagraphtype ,
5190 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5191 from .tl_set:N = \sparagraphfrom ,
5192 to .tl_set:N = \sparagraphto ,
5193 name .str_set:N = \sparagraphname
5194 }
5195 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5196 \tl_clear:N \sparagraphfrom

```

```

5197 \tl_clear:N \sparagraphto
5198 \str_clear:N \sparaagraphid
5199 \str_clear:N \sparaagraphtype
5200 \clist_clear:N \l__stex_statements_sparaagraph_for_clist
5201 \str_clear:N \sparaagraphname
5202 \keys_set:nn { stex / inlinepara }{ #1 }
5203 }
5204 \NewDocumentCommand \inlinepara { 0{} m } {
5205   \begingroup
5206   \__stex_statements_inlinepara_args:n{ #1 }
5207   \clist_set:No \l_tmpa_clist \sparaagraphtype
5208   \str_if_empty:NTF \sparaagraphid {
5209     \str_if_empty:NTF \sparaagraphname {
5210       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5211         \stex_ref_new_doc_target:n {}
5212       }
5213     } {
5214       \stex_ref_new_doc_target:n {}
5215     }
5216   } {
5217     \stex_ref_new_doc_target:n \sparaagraphid
5218   }
5219   \stex_if_smsmode:TF{
5220     \str_if_empty:NF \sparaagraphname {
5221       \stex_symdecl_do:nn{}{\sparaagraphname}
5222       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5223     }
5224   }{
5225     \seq_clear:N \l_tmpa_seq
5226     \clist_map_inline:Nn \l__stex_statements_sparaagraph_for_clist {
5227       \tl_if_empty:nF{ ##1 }{
5228         \stex_get_symbol:n { ##1 }
5229         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5230           \l_stex_get_symbol_uri_str
5231         }
5232       }
5233     }
5234     \exp_args:NNx
5235     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5236       \str_if_empty:NF \sparaagraphtype {
5237         \stex_annotate_invisible:nnn{type}{\sparaagraphtype}{}
5238       }
5239       \str_if_empty:NF \sparaagraphfrom {
5240         \stex_annotate_invisible:nnn{from}{\sparaagraphfrom}{}
5241       }
5242       \str_if_empty:NF \sparaagraphto {
5243         \stex_annotate_invisible:nnn{to}{\sparaagraphto}{}
5244       }
5245       \str_if_empty:NF \sparaagraphname {
5246         \stex_symdecl_do:nn{}{\sparaagraphname}
5247         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparaagraphname}
5248       }
5249       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5250         \clist_map_inline:Nn \l_tmpa_seq {

```

```

5251         \stex_ref_new_sym_target:n {##1}
5252     }
5253 }
5254 #2
5255 }
5256 }
5257 \endgroup
5258 \stex_smsmode_do:
5259 }
5260

```

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

```

5261 \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

```
5262 <*package>
5263 <@@=stex_sproof>
5264
5265 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5266
```

33.2 Proofs

We first define some keys for the proof environment.

```
5267 \keys_define:nn { stex / spf } {
5268   id          .str_set_x:N = \spfid,
5269   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5270   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5271   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5272   type        .str_set_x:N = \spftype,
5273   title       .tl_set:N    = \spftitle,
5274   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5275   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5276   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5277 }
5278 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5279   \str_clear:N \spfid
5280   \tl_clear:N \l__stex_sproof_spf_for_tl
5281   \tl_clear:N \l__stex_sproof_spf_from_tl
5282   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5283   \str_clear:N \spftype
5284   \tl_clear:N \spftitle
5285   \tl_clear:N \l__stex_sproof_spf_continues_tl
5286   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

⁸EdNOTE: need an implementation for L^AT_EX_ML

```

5287 \tl_clear:N \l__stex_sproof_spf_method_tl
5288 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5289 \keys_set:nn { stex / spf }{ #1 }
5290 }

```

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

```

5291 \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.

```

5292 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5293 \cs_new_protected:Npn \sproofnumber {
5294   \int_set:Nn \l_tmpa_int {1}
5295   \bool_while_do:nn {
5296     \int_compare_p:nNn {
5297       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5298     } > 0
5299   }{
5300     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5301     \int_incr:N \l_tmpa_int
5302   }
5303 }
5304 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5305   \int_set:Nn \l_tmpa_int {1}
5306   \bool_while_do:nn {
5307     \int_compare_p:nNn {
5308       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5309     } > 0
5310   }{
5311     \int_incr:N \l_tmpa_int
5312   }
5313   \int_compare:nNnF \l_tmpa_int = 1 {
5314     \int_decr:N \l_tmpa_int
5315   }
5316   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5317     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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


```

5362 }
5363 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5364   \input{sproof-finnish.ldf}
5365 }
5366 \clist_if_in:NnT \l_tmpa_clist {french}{
5367   \input{sproof-french.ldf}
5368 }
5369 \clist_if_in:NnT \l_tmpa_clist {russian}{
5370   \input{sproof-russian.ldf}
5371 }
5372 \makeatother
5373 }{}
5374 }

```

spfsketch

```

5375 \newcommand\spfsketch[2] [] {
5376   \beginingroup
5377   \let \premise \stex_proof_premise:
5378   \__stex_sproof_spf_args:n{#1}
5379   \stex_if_smsmode:TF {
5380     \str_if_empty:NF \spfid {
5381       \stex_ref_new_doc_target:n \spfid
5382     }
5383   }{
5384     \seq_clear:N \l_tmpa_seq
5385     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5386       \tl_if_empty:nF{ ##1 }{
5387         \stex_get_symbol:n { ##1 }
5388         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5389           \l_stex_get_symbol_uri_str
5390         }
5391       }
5392     }
5393     \exp_args:Nnx
5394     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5395       \str_if_empty:NF \spftype {
5396         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5397       }
5398       \clist_set:Nn \l_tmpa_clist \spftype
5399       \tl_set:Nn \l_tmpa_tl {
5400         \titleemph{
5401           \tl_if_empty:NTF \spftitle {
5402             \spf@proofsketch@kw
5403           }{
5404             \spftitle
5405           }
5406         }::~
5407       }
5408       \clist_map_inline:Nn \l_tmpa_clist {
5409         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5410           \tl_clear:N \l_tmpa_tl
5411         }
5412       }
5413       \str_if_empty:NF \spfid {

```

```

5414         \stex_ref_new_doc_target:n \spfid
5415     }
5416     \l_tmpa_tl #2 \sproofend
5417 }
5418 }
5419 \endgroup
5420 \stex_smsmode_do:
5421 }
5422

```

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

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

```

5423 \newenvironment{spfeq}[2][]{
5424   \__stex_sproof_spf_args:n{#1}
5425   \let \premise \stex_proof_premise:
5426   \stex_if_smsmode:TF {
5427     \str_if_empty:NF \spfid {
5428       \stex_ref_new_doc_target:n \spfid
5429     }
5430   }{
5431     \seq_clear:N \l_tmpa_seq
5432     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5433       \tl_if_empty:NF{ ##1 }{
5434         \stex_get_symbol:n { ##1 }
5435         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5436           \l_stex_get_symbol_uri_str
5437         }
5438       }
5439     }
5440     \exp_args:Nnnx
5441     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5442     \str_if_empty:NF \spftype {
5443       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5444     }
5445
5446     \clist_set:No \l_tmpa_clist \spftype
5447     \tl_clear:N \l_tmpa_tl
5448     \clist_map_inline:Nn \l_tmpa_clist {
5449       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5450         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5451       }
5452       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5453         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5454       }
5455     }
5456     \tl_if_empty:NTF \l_tmpa_tl {
5457       \__stex_sproof_spfeq_start:
5458     }{
5459       \l_tmpa_tl
5460     }{-#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


```

5461 \str_if_empty:NF \spfid {
5462   \stex_ref_new_doc_target:n \spfid
5463 }
5464 \begin{displaymath}\begin{array}{rc1l}
5465 }
5466 \stex_smsmode_do:
5467 }{
5468 \stex_if_smsmode:F {
5469   \end{array}\end{displaymath}
5470   \clist_set:No \l_tmpa_clist \spftype
5471   \tl_clear:N \l_tmpa_tl
5472   \clist_map_inline:Nn \l_tmpa_clist {
5473     \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5474       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5475     }
5476   }
5477   \tl_if_empty:NTF \l_tmpa_tl {
5478     \__stex_sproof_spfeq_end:
5479   }{
5480     \l_tmpa_tl
5481   }
5482   \end{stex_annotate_env}
5483 }
5484 }
5485
5486 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5487   \titleemph{
5488     \tl_if_empty:NTF \spftitle {
5489       \spf@proof@kw
5490     }{
5491       \spftitle
5492     }
5493   }:
5494 }
5495 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5496
5497 \newcommand\stexpatchspfeq[3] [] {
5498   \str_set:Nx \l_tmpa_str{ #1 }
5499   \str_if_empty:NTF \l_tmpa_str {
5500     \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5501     \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5502   }{
5503     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5504     \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5505   }
5506 }
5507

```

(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.

```

5508 \newenvironment{sproof}[2] []{

```

```

5509 \let \premise \stex_proof_premise:
5510 \intarray_gzero:N \l__stex_sproof_counter_intarray
5511 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5512 \__stex_sproof_spf_args:n{#1}
5513 \stex_if_smsmode:TF {
5514   \str_if_empty:NF \spfid {
5515     \stex_ref_new_doc_target:n \spfid
5516   }
5517 }{
5518   \seq_clear:N \l_tmpa_seq
5519   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5520     \tl_if_empty:NF{ ##1 }{
5521       \stex_get_symbol:n { ##1 }
5522       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5523         \l_stex_get_symbol_uri_str
5524       }
5525     }
5526   }
5527   \exp_args:Nnnx
5528   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5529   \str_if_empty:NF \spftype {
5530     \stex_annotate_invisible:nnn{type}{\spftype}{}
5531   }
5532
5533   \clist_set:No \l_tmpa_clist \spftype
5534   \tl_clear:N \l_tmpa_tl
5535   \clist_map_inline:Nn \l_tmpa_clist {
5536     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5537       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5538     }
5539     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5540       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5541     }
5542   }
5543   \tl_if_empty:NTF \l_tmpa_tl {
5544     \__stex_sproof_sproof_start:
5545   }{
5546     \l_tmpa_tl
5547   }{~#2}
5548   \str_if_empty:NF \spfid {
5549     \stex_ref_new_doc_target:n \spfid
5550   }
5551   \begin{description}
5552 }
5553 \stex_smsmode_do:
5554 }{
5555   \stex_if_smsmode:F{
5556     \end{description}
5557     \clist_set:No \l_tmpa_clist \spftype
5558     \tl_clear:N \l_tmpa_tl
5559     \clist_map_inline:Nn \l_tmpa_clist {
5560       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5561         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5562       }

```

```

5563     }
5564     \tl_if_empty:NTF \l_tmpa_tl {
5565         \__stex_sproof_sproof_end:
5566     }{
5567         \l_tmpa_tl
5568     }
5569     \end{stex_annotate_env}
5570 }
5571 }
5572
5573 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5574     \par\noindent\titleemph{
5575         \tl_if_empty:NTF \spftype {
5576             \spf@proof@kw
5577         }{
5578             \spftype
5579         }
5580     }:
5581 }
5582 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5583
5584 \newcommand\stexpatchproof[3] [] {
5585     \str_set:Nx \l_tmpa_str{ #1 }
5586     \str_if_empty:NTF \l_tmpa_str {
5587         \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5588         \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5589     }{
5590         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5591         \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5592     }
5593 }

```

\spfidea

```

5594 \newcommand\spfidea[2] []{
5595     \__stex_sproof_spf_args:n{#1}
5596     \titleemph{
5597         \tl_if_empty:NTF \spftype {Proof~Idea}{
5598             \spftype
5599         }:
5600     }~#2
5601     \sproofend
5602 }

```

(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

```

5603 \newenvironment{spfstep}[1] []{
5604     \__stex_sproof_spf_args:n{#1}
5605     \stex_if_smsmode:TF {

```

```

5606 \str_if_empty:NF \spfid {
5607   \stex_ref_new_doc_target:n \spfid
5608 }
5609 }{
5610   \@in@omtexttrue
5611   \seq_clear:N \l_tmpa_seq
5612   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5613     \tl_if_empty:NF{ ##1 }{
5614       \stex_get_symbol:n { ##1 }
5615       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5616         \l_stex_get_symbol_uri_str
5617       }
5618     }
5619   }
5620   \exp_args:Nnnx
5621   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5622   \str_if_empty:NF \spftype {
5623     \stex_annotate_invisible:nnn{type}{\spftype}{}
5624   }
5625   \clist_set:No \l_tmpa_clist \spftype
5626   \tl_set:Nn \l_tmpa_tl {
5627     \item[\sproofnumber]
5628     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5629   }
5630   \clist_map_inline:Nn \l_tmpa_clist {
5631     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5632       \tl_clear:N \l_tmpa_tl
5633     }
5634   }
5635   \l_tmpa_tl
5636   \tl_if_empty:NF \spftitle {
5637     {(\titleemph{\spftitle})\enspace}
5638   }
5639   \str_if_empty:NF \spfid {
5640     \stex_ref_new_doc_target:n \spfid
5641   }
5642 }
5643 \stex_smsmode_do:
5644 \ignorespacesandpars
5645 }{
5646   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5647     \__stex_sproof_inc_counter:
5648   }
5649   \stex_if_smsmode:F {
5650     \end{stex_annotate_env}
5651   }
5652 }

```

sproofcomment

```

5653 \newenvironment{sproofcomment}[1][]{
5654   \__stex_sproof_spf_args:n{#1}
5655   \clist_set:No \l_tmpa_clist \spftype
5656   \tl_set:Nn \l_tmpa_tl {
5657     \item[\sproofnumber]

```

```

5658 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5659 }
5660 \clist_map_inline:Nn \l_tmpa_clist {
5661   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5662     \tl_clear:N \l_tmpa_tl
5663   }
5664 }
5665 \l_tmpa_tl
5666 }{
5667   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5668     \__stex_sproof_inc_counter:
5669   }
5670 }

```

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.

```

5671 \newenvironment{subproof}[2][]{
5672   \__stex_sproof_spf_args:n{#1}
5673   \stex_if_smsmode:TF{
5674     \str_if_empty:NF \spfid {
5675       \stex_ref_new_doc_target:n \spfid
5676     }
5677   }{
5678     \seq_clear:N \l_tmpa_seq
5679     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5680       \tl_if_empty:nF{ ##1 }{
5681         \stex_get_symbol:n { ##1 }
5682         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5683           \l_stex_get_symbol_uri_str
5684         }
5685       }
5686     }
5687     \exp_args:Nnnx
5688     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5689     \str_if_empty:NF \spftype {
5690       \stex_annotate_invisible:nnn{type}{\spftype}{\}
5691     }
5692
5693     \clist_set:No \l_tmpa_clist \spftype
5694     \tl_set:Nn \l_tmpa_tl {
5695       \item[\sproofnumber]
5696       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5697     }
5698     \clist_map_inline:Nn \l_tmpa_clist {
5699       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5700         \tl_clear:N \l_tmpa_tl
5701       }
5702     }
5703     \l_tmpa_tl
5704     \tl_if_empty:NF \spftitle {
5705       {(\titleemph{\spftitle})\enspace}
5706     }

```

```

5707     {~#2}
5708     \str_if_empty:NF \spfid {
5709       \stex_ref_new_doc_target:n \spfid
5710     }
5711   }
5712   \__stex_sproof_add_counter:
5713   \stex_smsmode_do:
5714 }{
5715   \__stex_sproof_remove_counter:
5716   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5717     \__stex_sproof_inc_counter:
5718   }
5719   \stex_if_smsmode:F{
5720     \end{stex_annotate_env}
5721   }
5722 }

```

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

```

5723 \newenvironment{spfcases}[2][]{
5724   \tl_if_empty:nTF{#1}{
5725     \begin{subproof}[method=by-cases]{#2}
5726   }{
5727     \begin{subproof}[#1,method=by-cases]{#2}
5728   }
5729 }{
5730   \end{subproof}
5731 }

```

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

```

5732 \newenvironment{spfcase}[2][]{
5733   \__stex_sproof_spf_args:n{#1}
5734   \stex_if_smsmode:TF {
5735     \str_if_empty:NF \spfid {
5736       \stex_ref_new_doc_target:n \spfid
5737     }
5738   }{
5739     \seq_clear:N \l_tmpa_seq
5740     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5741       \tl_if_empty:nF{ ##1 }{
5742         \stex_get_symbol:n { ##1 }
5743         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5744           \l_stex_get_symbol_uri_str
5745         }
5746       }
5747     }
5748     \exp_args:Nnnx
5749     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5750     \str_if_empty:NF \spftype {
5751       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5752   }
5753   \clist_set:Nn \l_tmpa_clist \spftype
5754   \tl_set:Nn \l_tmpa_tl {
5755     \item[\sproofnumber]

```

```

5756     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5757   }
5758   \clist_map_inline:Nn \l_tmpa_clist {
5759     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5760       \tl_clear:N \l_tmpa_tl
5761     }
5762   }
5763   \l_tmpa_tl
5764   \tl_if_empty:nF{#2}{
5765     \titleemph{#2}:~
5766   }
5767 }
5768 \__stex_sproof_add_counter:
5769 \stex_smsmode_do:
5770 ){
5771   \__stex_sproof_remove_counter:
5772   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5773     \__stex_sproof_inc_counter:
5774   }
5775   \stex_if_smsmode:F{
5776     \clist_set:No \l_tmpa_clist \spftype
5777     \tl_set:Nn \l_tmpa_tl{\sproofend}
5778     \clist_map_inline:Nn \l_tmpa_clist {
5779       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5780         \tl_clear:N \l_tmpa_tl
5781       }
5782     }
5783     \l_tmpa_tl
5784     \end{stex_annotate_env}
5785   }
5786 }

```

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

```

5787 \newcommand\spfcasesketch[3][]{
5788   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5789 }

```

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.

```

5790 \keys_define:nn { stex / just }{
5791   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5792   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5793   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5794   args    .tl_set:N    = \l__stex_sproof_just_args_tl
5795 }

```

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

```
5796 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5797 \newcommand\stex_proof_premise:[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.

```
5798 \newcommand\justarg[2] [] {#2}
```

```
5799 \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

```
5800 <*package>
5801
5802 %%%%%%%%%%% others.dtx %%%%%%%%%%%
5803
5804 <@@=stex_others>
    Warnings and error messages
5805 % None

\MSC Math subject classifier

5806 \NewDocumentCommand \MSC {m} {
5807 % TODO
5808 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5809 \@ifpackageloaded{tikzinput}{
5810 \RequirePackage{stex-tikzinput}
5811 }{}
5812 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
5813 <*package>
5814 <@@=stex_modules>
5815
5816 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5817
5818 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5819 \begingroup
5820 \stex_module_setup:nn{
5821   ns=\c_stex_metatheory_ns_str,
5822   meta=NONE
5823 }{Metatheory}
5824 \stex_reactivate_macro:N \symdecl
5825 \stex_reactivate_macro:N \notation
5826 \stex_reactivate_macro:N \symdef
5827 \ExplSyntaxOff
5828 \csname stex_suppress_html:n\endcsname{
5829   % is-a (a:A, a \in A, a is an A, etc.)
5830   \symdecl{isa}[args=ai]
5831   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5832   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5833   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5834
5835   % bind (\forall, \Pi, \lambda etc.)
5836   \symdecl{bind}[args=Bi]
5837   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5838   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5839   \notation{bind}[depfun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5840
5841   % implicit bind
5842   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5843
5844   % dummy variable
5845   \symdecl{dummyvar}
5846   \notation{dummyvar}[underscore]{\comp\_}
5847   \notation{dummyvar}[dot]{\comp\cdot}
```

```

5848 \notation{dummyvar}[dash]{\comp{\rm --}}
5849
5850 %fromto (function space, Hom-set, implication etc.)
5851 \symdecl{fromto}[args=ai]
5852 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5853 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5854
5855 % mapto (lambda etc.)
5856 \symdecl{mapto}[args=Bi]
5857 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5858 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5859 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5860
5861 % function/operator application
5862 \symdecl{apply}[args=ia]
5863 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5864 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5865
5866 % ‘‘type’’ of all collections (sets,classes,types,kinds)
5867 \symdecl{metacollection}
5868 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5869 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5870
5871 % collection of propositions/booleans/truth values
5872 \symdecl{prop}[name=proposition]
5873 \notation{prop}[prop]{\comp{\rm prop}}
5874 \notation{prop}[B00L]{\comp{\rm B00L}}
5875
5876 % sequences
5877 \symdecl{seqtype}[args=1]
5878 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5879
5880 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{1}_{#2}}
5881 \notation{sequence-index}[ui,prec=nobrackets]{#{1}^{\comp\ast}}
5882
5883 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5884 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5885 \symdef{aseqfromtovia}[args=aai,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5886
5887 % letin (‘‘let’’, local definitions, variable substitution)
5888 \symdecl{letin}[args=bii]
5889 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}#2\; \comp{\rm in}}\;#3}
5890 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5891 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5892
5893 % structures
5894 \symdecl*{module-type}[args=1]
5895 \notation{module-type}{\mathtt{MOD} #1}
5896 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5897 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5898
5899 }
5900 \ExplSyntaxOn
5901 \stex_add_to_current_module:n{

```

```

5902 \let\nappa\apply
5903 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5904 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5905 \def\livar{\csname sequence-index\endcsname[li]}
5906 \def\uivar{\csname sequence-index\endcsname[ui]}
5907 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5908 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5909 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5910 }
5911 \__stex_modules_end_module:
5912 \endgroup
5913 </package>

```

Chapter 36

Tikzinput Implementation

```
5914 <*package>
5915
5916 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5917
5918 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5919 \RequirePackage{l3keys2e}
5920
5921 \keys_define:nn { tikzinput } {
5922   image .bool_set:N = \c_tikzinput_image_bool,
5923   image .default:n = false ,
5924   unknown .code:n = {}
5925 }
5926
5927 \ProcessKeysOptions { tikzinput }
5928
5929 \bool_if:NTF \c_tikzinput_image_bool {
5930   \RequirePackage{graphicx}
5931
5932   \providecommand\usetikzlibrary[]{}
5933   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
5934 }{
5935   \RequirePackage{tikz}
5936   \RequirePackage{standalone}
5937
5938   \newcommand \tikzinput [2] [] {
5939     \setkeys{Gin}{#1}
5940     \ifx \Gin@ewidth \Gin@exclamation
5941       \ifx \Gin@eheight \Gin@exclamation
5942         \input { #2 }
5943       \else
5944         \resizebox{!}{ \Gin@eheight }{
5945           \input { #2 }
5946         }
5947       \fi
5948     \else
5949       \ifx \Gin@eheight \Gin@exclamation
5950         \resizebox{ \Gin@ewidth }{!}{
5951           \input { #2 }
```

```

5952     }
5953     \else
5954         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
5955             \input { #2 }
5956         }
5957     \fi
5958 \fi
5959 }
5960 }
5961
5962 \newcommand \ctikzinput [2] [] {
5963     \begin{center}
5964         \tikzinput [1] {#2}
5965     \end{center}
5966 }
5967
5968 \@ifpackageloaded{stex}{
5969     \RequirePackage{stex-tikzinput}
5970 }{}
5971
5972 </package>
5973 <*stex>
5974 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
5975 \RequirePackage{stex}
5976 \RequirePackage{tikzinput}
5977
5978 \newcommand\mhtikzinput [2] [] {%
5979     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
5980     \stex_in_repository:nn\Gin@mhrepos{
5981         \tikzinput[#1]{\mhpath{##1}{#2}}
5982     }
5983 }
5984 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
5985 </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.

```
5986 \*cls)
5987 \@@=document_structure)
5988 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
5989 \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`

```
5990 \keys_define:nn{ document-structure / pkg }{
5991   class      .str_set_x:N = \c_document_structure_class_str,
5992   minimal    .bool_set:N = \c_document_structure_minimal_bool,
5993   report     .code:n      = {
5994     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
5995     \str_set:Nn \c_document_structure_class_str {report}
5996   },
5997   book       .code:n      = {
5998     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
5999     \str_set:Nn \c_document_structure_class_str {book}
6000   },
6001   bookpart   .code:n      = {
6002     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6003     \str_set:Nn \c_document_structure_class_str {book}
6004     \str_set:Nn \c_document_structure_topsect_str {chapter}
6005   },
```

```

6006 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6007 unknown    .code:n      = {
6008   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6009 }
6010 }
6011 \ProcessKeysOptions{ document-structure / pkg }
6012 \str_if_empty:NT \c_document_structure_class_str {
6013   \str_set:Nn \c_document_structure_class_str {article}
6014 }
6015 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6016   {\c_document_structure_class_str}
6017

```

37.3 Beefing up the document environment

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

```

6018 \RequirePackage{document-structure}
6019 \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

```

6020 \keys_define:nn { document-structure / document }{
6021   id .str_set_x:N = \c_document_structure_document_id_str
6022 }
6023 \let\__document_structure_orig_document=\document
6024 \renewcommand{\document}[1][]{
6025   \keys_set:nn{ document-structure / document }{ #1 }
6026   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6027   \__document_structure_orig_document
6028 }

```

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

```

6029 }
6030 \</cls>

```

37.4 Implementation: document-structure Package

```

6031 \<*package>
6032 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6033 \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


```

6034
6035 \keys_define:nn{ document-structure / pkg }{
6036   class      .str_set_x:N = \c_document_structure_class_str,
6037   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6038   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6039 }
6040 \ProcessKeysOptions{ document-structure / pkg }
6041 \str_if_empty:NT \c_document_structure_class_str {
6042   \str_set:Nn \c_document_structure_class_str {article}
6043 }
6044 \str_if_empty:NT \c_document_structure_topsect_str {
6045   \str_set:Nn \c_document_structure_topsect_str {section}
6046 }

```

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

```

6047 \RequirePackage{xspace}
6048 \RequirePackage{comment}
6049 \AddToHook{begindocument}{
6050   \ltx@ifpackageloaded{babel}{
6051     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6052     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6053       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6054     }
6055   }{}
6056 }

```

`\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.

```

6057 \int_new:N \l_document_structure_section_level_int
6058 \str_case:VnF \c_document_structure_topsect_str {
6059   {part}}{
6060     \int_set:Nn \l_document_structure_section_level_int {0}
6061   }
6062   {chapter}}{
6063     \int_set:Nn \l_document_structure_section_level_int {1}
6064   }
6065 }{
6066   \str_case:VnF \c_document_structure_class_str {
6067     {book}}{
6068       \int_set:Nn \l_document_structure_section_level_int {0}
6069     }
6070     {report}}{
6071       \int_set:Nn \l_document_structure_section_level_int {0}
6072     }
6073   }{
6074     \int_set:Nn \l_document_structure_section_level_int {2}
6075   }
6076 }

```

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

```
6077 \def\current@section@level{document}%
6078 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6079 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6080 \cs_new_protected:Npn \skipomgroup {
6081   \ifcase\l_document_structure_section_level_int
6082   \or\stepcounter{part}
6083   \or\stepcounter{chapter}
6084   \or\stepcounter{section}
6085   \or\stepcounter{subsection}
6086   \or\stepcounter{subsubsection}
6087   \or\stepcounter{paragraph}
6088   \or\stepcounter{subparagraph}
6089   \fi
6090 }
```

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

`blindfragment`

```
6091 \newcommand\at@begin@blindomgroup[1]{%
6092 \newenvironment{blindfragment}
6093 {
6094   \int_incr:N\l_document_structure_section_level_int
6095   \at@begin@blindomgroup\l_document_structure_section_level_int
6096 }{}}
```

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

```
6097 \newcommand\omgroup@nonum[2]{
6098   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6099   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6100 }
```

(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.

```
6101 \newcommand\omgroup@num[2]{
```

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

```

6102 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6103   \@nameuse{#1}{#2}
6104 }{
6105   \cs_if_exist:NTF\rdfmata@sectioning{
6106     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6107   }{
6108     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6109   }
6110 }
6111 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6112 }

```

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

sfragment

```

6113 \keys_define:nn { document-structure / omgroup }{
6114   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6115   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6116   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6117   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6118   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6119   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6120   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6121   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6122   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6123   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6124 }
6125 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6126   \str_clear:N \l__document_structure_omgroup_id_str
6127   \str_clear:N \l__document_structure_omgroup_date_str
6128   \clist_clear:N \l__document_structure_omgroup_creators_clist
6129   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6130   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6131   \tl_clear:N \l__document_structure_omgroup_type_tl
6132   \tl_clear:N \l__document_structure_omgroup_short_tl
6133   \tl_clear:N \l__document_structure_omgroup_display_tl
6134   \tl_clear:N \l__document_structure_omgroup_intro_tl
6135   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6136   \keys_set:nn { document-structure / omgroup } { #1 }
6137 }

```

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.

```

6138 \newif\if@mainmatter\@mainmattertrue
6139 \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.

```

6140 \keys_define:nn { document-structure / sectioning }{
6141   name .str_set_x:N = \l__document_structure_sect_name_str ,
6142   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6143   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6144   clear .default:n = {true} ,
6145   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6146   num      .default:n      = {true}
6147 }
6148 \cs_new_protected:Nn \__document_structure_sect_args:n {
6149   \str_clear:N \l__document_structure_sect_name_str
6150   \str_clear:N \l__document_structure_sect_ref_str
6151   \bool_set_false:N \l__document_structure_sect_clear_bool
6152   \bool_set_false:N \l__document_structure_sect_num_bool
6153   \keys_set:nn { document-structure / sectioning } { #1 }
6154 }
6155 \newcommand\omdoc@sectioning[3][]{
6156   \__document_structure_sect_args:n {#1}
6157   \let\omdoc@sect@name\l__document_structure_sect_name_str
6158   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6159   \if@mainmatter% numbering not overridden by frontmatter, etc.
6160     \bool_if:NTF \l__document_structure_sect_num_bool {
6161       \omgroup@num{#2}{#3}
6162     }{
6163       \omgroup@nonum{#2}{#3}
6164     }
6165     \def\current@section@level{\omdoc@sect@name}
6166   \else
6167     \omgroup@nonum{#2}{#3}
6168   \fi
6169 }% 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.

```

6170 \newcommand\omgroup@redefine@addtocontents[1]{%
6171 %\edef\__document_structureimport{#1}%
6172 %\@for\@I:=\__document_structureimport\do{%
6173 %\edef\@path{\csname module@\@I @path\endcsname}%
6174 %\@ifundefined{tf@toc}\relax%
6175 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6176 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6177 %\def\addcontentsline##1##2##3{%
6178 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6179 %\else% hyperref.sty not loaded
6180 %\def\addcontentsline##1##2##3{%
6181 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6182 %\fi
6183 }% 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.

```

6184 \newenvironment{sfragment}[2][]{% keys, title
6185 {
6186   \__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.

```

6187   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6188     \omgroup@redefine@addtocontents{
6189       \@ifundefined{module@id}\used@modules%

```

```

6190     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6191   }
6192 }

```

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

```

6193 \int_incr:N\l_document_structure_section_level_int
6194 \ifcase\l_document_structure_section_level_int
6195   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6196   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6197   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6198   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6199   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6200   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6201   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6202 \fi
6203 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6204 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6205   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6206 }
6207 }% for customization
6208 {}

```

and finally, we localize the sections

```

6209 \newcommand\omdoc@part@kw{Part}
6210 \newcommand\omdoc@chapter@kw{Chapter}
6211 \newcommand\omdoc@section@kw{Section}
6212 \newcommand\omdoc@subsection@kw{Subsection}
6213 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6214 \newcommand\omdoc@paragraph@kw{paragraph}
6215 \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`

```

6216 \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).

```

6217 \cs_if_exist:NTF\frontmatter{
6218   \let\__document_structure_orig_frontmatter\frontmatter
6219   \let\frontmatter\relax
6220 }{
6221   \tl_set:Nn\__document_structure_orig_frontmatter{
6222     \clearpage
6223     \@mainmatterfalse
6224     \pagenumbering{roman}

```

```

6225 }
6226 }
6227 \cs_if_exist:NTF\backmatter{
6228   \let\__document_structure_orig_backmatter\backmatter
6229   \let\backmatter\relax
6230 }{
6231   \tl_set:Nn\__document_structure_orig_backmatter{
6232     \clearpage
6233     \@mainmatterfalse
6234     \pagenumbering{roman}
6235   }
6236 }

```

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.

```

6237 \newenvironment{frontmatter}{
6238   \__document_structure_orig_frontmatter
6239 }{
6240   \cs_if_exist:NTF\mainmatter{
6241     \mainmatter
6242   }{
6243     \clearpage
6244     \@mainmattertrue
6245     \pagenumbering{arabic}
6246   }
6247 }

```

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

```

6248 \newenvironment{backmatter}{
6249   \__document_structure_orig_backmatter
6250 }{
6251   \cs_if_exist:NTF\mainmatter{
6252     \mainmatter
6253   }{
6254     \clearpage
6255     \@mainmattertrue
6256     \pagenumbering{arabic}
6257   }
6258 }

```

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

```

6259 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6260 \def \c__document_structure_document_str{document}
6261 \newcommand\afterprematurestop{}
6262 \def\prematurestop@endomgroup{
6263   \unless\ifx\@currenvir\c__document_structure_document_str
6264     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6265     \expandafter\prematurestop@endomgroup

```

```

6266 \fi
6267 }
6268 \providecommand\prematurestop{
6269 \message{Stopping~sTeX~processing~prematurely}
6270 \prematurestop@endomgroup
6271 \afterprematurestop
6272 \end{document}
6273 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

6274 \RequirePackage{etoolbox}
6275 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6276 \newrobustcmd\useSGvar[1]{%
6277 \@ifundefined{sTeX@Gvar@#1}
6278 {\PackageError{document-structure}
6279 {The sTeX Global variable #1 is undefined}
6280 {set it with \protect\setSGvar}}
6281 \@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.

```

6282 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6283 \@ifundefined{sTeX@Gvar@#1}
6284 {\PackageError{document-structure}
6285 {The sTeX Global variable #1 is undefined}
6286 {set it with \protect\setSGvar}}
6287 {\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.

```
6288 \*cls)
6289 \@@=notesslides)
6290 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6291 \RequirePackage{13keys2e}
6292
6293 \keys_define:nn{notesslides / cls}{
6294   class .code:n = {
6295     \PassOptionsToClass{\CurrentOption}{document-structure}
6296     \str_if_eq:nnT{#1}{book}{
6297       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6298     }
6299     \str_if_eq:nnT{#1}{report}{
6300       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6301     }
6302   },
6303   notes .bool_set:N = \c__notesslides_notes_bool ,
6304   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6305   unknown .code:n = {
6306     \PassOptionsToClass{\CurrentOption}{document-structure}
6307     \PassOptionsToClass{\CurrentOption}{beamer}
6308     \PassOptionsToPackage{\CurrentOption}{notesslides}
6309   }
6310 }
6311 \ProcessKeysOptions{ notesslides / cls }
6312 \bool_if:NTF \c__notesslides_notes_bool {
6313   \PassOptionsToPackage{notes=true}{notesslides}
6314 }{
6315   \PassOptionsToPackage{notes=false}{notesslides}
6316 }
6317 \</cls)
```


now we do the same for the notesslides package.

```

6318 <*package>
6319 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6320 \RequirePackage{13keys2e}
6321
6322 \keys_define:nn{notesslides / pkg}{
6323   topsect      .str_set_x:N = \c__notesslides_topsect_str,
6324   defaulttopsect .str_set_x:N = \c__notesslides_defaulttopsec_str,
6325   notes        .bool_set:N = \c__notesslides_notes_bool ,
6326   slides       .code:n      = { \bool_set_false:N \c__notesslides_notes_bool },
6327   sectocframes .bool_set:N = \c__notesslides_sectocframes_bool ,
6328   frameimages  .bool_set:N = \c__notesslides_frameimages_bool ,
6329   fiboxed      .bool_set:N = \c__notesslides_fiboxed_bool ,
6330   nopproblems  .bool_set:N = \c__notesslides_nopproblems_bool,
6331   unknown      .code:n      = {
6332     \PassOptionsToClass{\CurrentOption}{stex}
6333     \PassOptionsToClass{\CurrentOption}{tikzinput}
6334   }
6335 }
6336 \ProcessKeysOptions{ notesslides / pkg }
6337 \newif\ifnotes
6338 \bool_if:NTF \c__notesslides_notes_bool {
6339   \notesttrue
6340 }{
6341   \notesfalse
6342 }
6343

```

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

```

6344 \str_if_empty:NTF \c__notesslides_topsect_str {
6345   \str_set_eq:NN \__notesslides_topsect \c__notesslides_defaulttopsec_str
6346 }{
6347   \str_set_eq:NN \__notesslides_topsect \c__notesslides_topsect_str
6348 }
6349 </package>

```

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

```

6350 <*cls>
6351 \bool_if:NTF \c__notesslides_notes_bool {
6352   \LoadClass{document-structure}
6353 }{
6354   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6355   \newcounter{Item}
6356   \newcounter{paragraph}
6357   \newcounter{subparagraph}
6358   \newcounter{Hfootnote}
6359   \RequirePackage{document-structure}
6360 }

```

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

```

6361 \RequirePackage{notesslides}
6362 </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.

```

6363 \*package>
6364 \bool_if:NT \c__notesslides_notes_bool {
6365   \RequirePackage{a4wide}
6366   \RequirePackage{marginnote}
6367   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6368   \RequirePackage{mdframed}
6369   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6370   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6371 }
6372 \RequirePackage{stex-tikzinput}
6373 \RequirePackage{etoolbox}
6374 \RequirePackage{amssymb}
6375 \RequirePackage{amsmath}
6376 \RequirePackage{comment}
6377 \RequirePackage{textcomp}
6378 \RequirePackage{url}
6379 \RequirePackage{graphicx}
6380 \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

```

6381 \bool_if:NT \c__notesslides_notes_bool {
6382   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6383 }
6384
6385
6386 \NewDocumentCommand \libusetheme {0{} m} {
6387   \bool_if:NTF \c__notesslides_notes_bool {
6388     \libusepackage[#1]{beamernotestheme#2}
6389   }{
6390     \libusepackage[#1]{beamertheme#2}
6391   }
6392 }

```

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

```

6393 \newcounter{slide}
6394 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6395 \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.

```

6396 \bool_if:NTF \c__notesslides_notes_bool {
6397   \renewenvironment{note}{\ignorespaces}{}
6398 }{
6399   \excludecomment{note}
6400 }

```

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.

```

6401 \bool_if:NT \c__notesslides_notes_bool {
6402   \newlength{\slideframewidth}
6403   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6404 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6405   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6406     \bool_set_true:N #1
6407   }{
6408     \bool_set_false:N #1
6409   }
6410 }
6411 \keys_define:nn{notesslides / frame}{
6412   label .str_set_x:N = \l__notesslides_frame_label_str,
6413   allowframebreaks .code:n = {
6414     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6415   },
6416   allowdisplaybreaks .code:n = {
6417     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6418   },
6419   fragile .code:n = {
6420     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6421   },
6422   shrink .code:n = {
6423     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6424   },
6425   squeeze .code:n = {
6426     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6427   },
6428   t .code:n = {
6429     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6430   },
6431 }
6432 \cs_new_protected:Nn \__notesslides_frame_args:n {
6433   \str_clear:N \l__notesslides_frame_label_str
6434   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6435   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6436   \bool_set_true:N \l__notesslides_frame_fragile_bool
6437   \bool_set_true:N \l__notesslides_frame_shrink_bool
6438   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6439   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6440     \keys_set:nn { notesslides / frame }{ #1 }
6441   }

```

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

```

6442   \renewenvironment{frame}[1][]{
6443     \__notesslides_frame_args:n{#1}
6444     \sffamily
6445     \stepcounter{slide}
6446     \def\@currentlabel{\theslide}
6447     \str_if_empty:NF \l__notesslides_frame_label_str {
6448       \label{\l__notesslides_frame_label_str}
6449     }

```

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

```

6450     \def\itemize@level{outer}
6451     \def\itemize@outer{outer}
6452     \def\itemize@inner{inner}
6453     \renewcommand\newpage{\addtocounter{framenumber}{1}}
6454     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6455     \renewenvironment{itemize}{
6456       \ifx\itemize@level\itemize@outer
6457         \def\itemize@label{$\rhd$}
6458       \fi
6459       \ifx\itemize@level\itemize@inner
6460         \def\itemize@label{$\scriptstyle\rhd$}
6461       \fi
6462       \begin{list}
6463         {\itemize@label}
6464         {\setlength{\labelsep}{.3em}
6465          \setlength{\labelwidth}{.5em}
6466          \setlength{\leftmargin}{1.5em}
6467         }
6468       \edef\itemize@level{\itemize@inner}
6469     }{
6470       \end{list}
6471     }

```

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

```

6472     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6473   }{
6474     \medskip\miko@slidelabel\end{mdframed}
6475   }

```

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

`\frametitle`

```

6476   \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6477 }

```

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

EdN:15

`\pause`

```

15
6478 \bool_if:NT \c__notesslides_notes_bool {
6479   \newcommand\pause{}
6480 }

```

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

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

nparagraph

```
6481 \bool_if:NTF \c__notesslides_notes_bool {
6482   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6483 }{
6484   \excludecomment{nparagraph}
6485 }
```

nfragment

```
6486 \bool_if:NTF \c__notesslides_notes_bool {
6487   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6488 }{
6489   \excludecomment{nfragment}
6490 }
```

ndefinition

```
6491 \bool_if:NTF \c__notesslides_notes_bool {
6492   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6493 }{
6494   \excludecomment{ndefinition}
6495 }
```

nassertion

```
6496 \bool_if:NTF \c__notesslides_notes_bool {
6497   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6498 }{
6499   \excludecomment{nassertion}
6500 }
```

nsproof

```
6501 \bool_if:NTF \c__notesslides_notes_bool {
6502   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6503 }{
6504   \excludecomment{nsproof}
6505 }
```

nexample

```
6506 \bool_if:NTF \c__notesslides_notes_bool {
6507   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6508 }{
6509   \excludecomment{nexample}
6510 }
```

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

```
6511 \def\inputref@preskip{\smallskip}
6512 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6513 \let\orig@inputref\inputref
6514 \def\inputref{\@ifstar\ninputref\orig@inputref}
6515 \newcommand\ninputref[2][]{
6516   \bool_if:NT \c__notesslides_notes_bool {
6517     \orig@inputref[#1]{#2}
6518   }
6519 }
```

(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>}`.

```
6520 \newlength{\slidelogoheight}
6521
6522 \bool_if:NTF \c__notesslides_notes_bool {
6523   \setlength{\slidelogoheight}{.4cm}
6524 }{
6525   \setlength{\slidelogoheight}{1cm}
6526 }
6527 \newsavebox{\slidelogo}
6528 \sbox{\slidelogo}{\sTeX}
6529 \newrobustcmd{\setslidelogo}[1]{
6530   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6531 }
```

(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.

```
6532 \def\source{Michael Kohlhase}% customize locally
6533 \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.

```
6534 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6535 \newsavebox{\cclogo}
6536 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6537 \newif\ifcchref\cchreffalse
6538 \AtBeginDocument{
6539   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6540 }
6541 \def\licensing{
6542   \ifcchref
```

```

6543     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6544 \else
6545     {\usebox{\cclogo}}
6546 \fi
6547 }
6548 \newrobustcmd{\setlicensing}[2][]{
6549   \def\@url{\#1}
6550   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
6551   \ifx\@url\@empty
6552     \def\licensing{\usebox{\cclogo}}
6553   \else
6554     \def\licensing{
6555       \ifcchref
6556         \href{\#1}{\usebox{\cclogo}}
6557       \else
6558         {\usebox{\cclogo}}
6559       \fi
6560     }
6561   \fi
6562 }

```

(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

```

6563 \newrobustcmd\miko@slidelabel{
6564   \vbox to \slidelogoheight{
6565     \vss\hbox to \slidewidth
6566     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6567   }
6568 }

```

(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.

```

6569 \def\Gin@mhrepos{}
6570 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{\#1}}
6571 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
6572 \newrobustcmd\frameimage[2][]{
6573   \stepcounter{slide}
6574   \bool_if:NT \c__notesslides_frameimages_bool {
6575     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
6576     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6577     \begin{center}
6578       \bool_if:NTF \c__notesslides_fboxed_bool {
6579         \fbox{
6580           \ifx\Gin@ewidth\@empty
6581             \ifx\Gin@mhrepos\@empty
6582               \mhgraphics[width=\slidewidth,\#1]{\#2}
6583             \else

```

¹⁶EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

6584         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6585     \fi
6586 \else% Gin@ewidth empty
6587     \ifx\Gin@mhrepos\@empty
6588         \mhgraphics[#1]{#2}
6589     \else
6590         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6591     \fi
6592 \fi% Gin@ewidth empty
6593 }
6594 }{
6595     \ifx\Gin@ewidth\@empty
6596     \ifx\Gin@mhrepos\@empty
6597         \mhgraphics[width=\slidewidth,#1]{#2}
6598     \else
6599         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6600     \fi
6601     \ifx\Gin@mhrepos\@empty
6602         \mhgraphics[#1]{#2}
6603     \else
6604         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6605     \fi
6606 \fi% Gin@ewidth empty
6607 }
6608 \end{center}
6609 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6610 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6611 }
6612 } % 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.

```

6613 \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

```

6614 \AddToHook{begindocument}{
6615     \definecolor{green}{rgb}{0,.5,0}
6616     \definecolor{purple}{cmyk}{.3,1,0,.17}
6617 }

```

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`.

```

6618 % \def\STpresent#1{\textcolor{blue}{#1}}
6619 \def\defemph#1{\textcolor{magenta}{#1}}
6620 \def\symrefemph#1{\textcolor{cyan}{#1}}
6621 \def\compemph#1{\textcolor{blue}{#1}}
6622 \def\titleemph#1{\textcolor{blue}{#1}}
6623 \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.

```

6624 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6625 \def\smalltextwarning{
6626   \pgfuseimage{miko@small@dbend}
6627   \xspace
6628 }
6629 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6630 \newrobustcmd\textwarning{
6631   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6632   \xspace
6633 }
6634 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6635 \newrobustcmd\bigtextwarning{
6636   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6637   \xspace
6638 }

```

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

```

6639 \newrobustcmd\putgraphicsat[3]{
6640   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6641 }
6642 \newrobustcmd\putat[2]{
6643   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6644 }

```

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`.

```

6645 \bool_if:NT \c__notesslides_sectocframes_bool {
6646   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6647     \newcounter{chapter}\counterwithin*{section}{chapter}
6648   }{
6649     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6650       \newcounter{chapter}\counterwithin*{section}{chapter}
6651     }
6652   }
6653 }

```

`\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
6654 \def\part@prefix{}
6655 \@ifpackageloaded{document-structure}{}{
6656   \str_case:VnF \__notesslidesstopsect {
6657     {part}{
6658       \int_set:Nn \l_document_structure_section_level_int {0}
6659       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6660     \def\part@prefix{\arabic{chapter}.}
6661   }
6662   {chapter}{
6663     \int_set:Nn \l_document_structure_section_level_int {1}
6664     \def\thesection{\arabic{chapter}.\arabic{section}}
6665     \def\part@prefix{\arabic{chapter}.}
6666   }
6667 }{
6668   \int_set:Nn \l_document_structure_section_level_int {2}
6669   \def\part@prefix{}
6670 }
6671 }
6672
6673 \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

```

6674 \renewenvironment{sfragment}[2][]{
6675   \_document_structure_omgroup_args:n { #1 }
6676   \int_incr:N \l_document_structure_section_level_int
6677   \bool_if:NT \c__notesslides_sectocframes_bool {
6678     \stepcounter{slide}
6679     \begin{frame}[noframenumbering]
6680     \vfill\Large\centering
6681     \red{
6682       \ifcase\l_document_structure_section_level_int\or
6683         \stepcounter{part}
6684         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6685         \def\currentsectionlevel{\omdoc@part@kw}
6686       \or
6687         \stepcounter{chapter}
6688         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6689         \def\currentsectionlevel{\omdoc@chapter@kw}
6690       \or
6691         \stepcounter{section}
6692         \def\_notesslideslabel{\part@prefix\arabic{section}}
6693         \def\currentsectionlevel{\omdoc@section@kw}
6694       \or
6695         \stepcounter{subsection}
6696         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6697         \def\currentsectionlevel{\omdoc@subsection@kw}
6698       \or
6699         \stepcounter{subsubsection}
6700         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6701         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6702       \or
6703         \stepcounter{paragraph}
6704         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6705         \def\currentsectionlevel{\omdoc@paragraph@kw}
6706       \else
6707         \def\_notesslideslabel{}

```

```

6708         \def\currentsectionlevel{\omdoc@paragraph@kw}
6709         \fi% end ifcase
6710         \_notesslideslabel%\sref@label@id\_notesslideslabel
6711         \quad #2%
6712     }%
6713     \vfill%
6714     \end{frame}%
6715 }
6716 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6717     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6718 }
6719 }{}
6720 }

```

We set up a beamer template for theorems like ams style, but without a block environment.

```

6721 \def\inserttheorembodyfont{\normalfont}
6722 %\bool_if:NF \c__notesslides_notes_bool {
6723 % \defbeamertemplate{theorem begin}{miko}
6724 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6725 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6726 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6727 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6728 % \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.

```

6729 % \expandafter\def\csname Parent2\endcsname{}
6730 %}
6731
6732 \AddToHook{begindocument}{% this does not work for some reason
6733     \setbeamertemplate{theorems}[ams style]
6734 }
6735 \bool_if:NT \c__notesslides_notes_bool {
6736     \renewenvironment{columns}[1][ ]{%
6737         \par\noindent%
6738         \begin{minipage}%
6739             \slidewidth\centering\leavevmode%
6740     }{%
6741         \end{minipage}\par\noindent%
6742     }%
6743     \newsavebox\columnbox%
6744     \renewenvironment<>{column}[2][ ]{%
6745         \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6746     }{%
6747         \end{minipage}\end{lrbox}\usebox\columnbox%
6748     }%
6749 }
6750 \bool_if:NTF \c__notesslides_noproblems_bool {
6751     \newenvironment{problems}{}{}
6752 }{
6753     \excludecomment{problems}
6754 }

```

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.

```

6755 \gdef\printexcursions{}
6756 \newcommand\excursionref[2]{% label, text
6757   \bool_if:NT \c__notesslides_notes_bool {
6758     \begin{sparagraph}[title=Excursion]
6759       #2 \sref[fallback=the appendix]{#1}.
6760     \end{sparagraph}
6761   }
6762 }
6763 \newcommand\activate@excursion[2][]{
6764   \gappto\printexcursions{\inputref{#1}{#2}}
6765 }
6766 \newcommand\excursion[4][]{% repos, label, path, text
6767   \bool_if:NT \c__notesslides_notes_bool {
6768     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6769   }
6770 }

```

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

`\excursiongroup`

```

6771 \keys_define:nn{notesslides / excursiongroup }{
6772   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6773   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6774   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6775 }
6776 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6777   \tl_clear:N \l__notesslides_excursion_intro_tl
6778   \str_clear:N \l__notesslides_excursion_id_str
6779   \str_clear:N \l__notesslides_excursion_mhrepos_str
6780   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6781 }
6782 \newcommand\excursiongroup[1][]{
6783   \__notesslides_excursion_args:n{ #1 }
6784   \ifdefempty\printexcursions{}% only if there are excursions
6785     {\begin{note}
6786       \begin{sfragment}[#1]{Excursions}%
6787         \ifdefempty\l__notesslides_excursion_intro_tl{\{
6788           \inputref[\l__notesslides_excursion_mhrepos_str]{
6789             \l__notesslides_excursion_intro_tl
6790           }
6791         }
6792         \printexcursions%
6793       \end{sfragment}
6794     \end{note}}
6795 }
6796 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6797 \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.

```
6798 <*package>
6799 <@@=problems>
6800 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6801 \RequirePackage{l3keys2e,stex}
6802
6803 \keys_define:nn { problem / pkg }{
6804   notes      .default:n    = { true },
6805   notes      .bool_set:N   = \c__problems_notes_bool,
6806   gnotes     .default:n    = { true },
6807   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6808   hints      .default:n    = { true },
6809   hints      .bool_set:N   = \c__problems_hints_bool,
6810   solutions  .default:n    = { true },
6811   solutions  .bool_set:N   = \c__problems_solutions_bool,
6812   pts        .default:n    = { true },
6813   pts        .bool_set:N   = \c__problems_pts_bool,
6814   min        .default:n    = { true },
6815   min        .bool_set:N   = \c__problems_min_bool,
6816   boxed      .default:n    = { true },
6817   boxed      .bool_set:N   = \c__problems_boxed_bool,
6818   unknown    .code:n       = {}
6819 }
6820 \newif\ifsolutions
6821
6822 \ProcessKeysOptions{ problem / pkg }
6823 \bool_if:NTF \c__problems_solutions_bool {
6824   \solutionstrue
6825 }{
6826   \solutionsfalse
6827 }
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
6828 \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.

```
6829 \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.

```
6830 \def\prob@problem@kw{Problem}
6831 \def\prob@solution@kw{Solution}
6832 \def\prob@hint@kw{Hint}
6833 \def\prob@note@kw{Note}
6834 \def\prob@gnote@kw{Grading}
6835 \def\prob@pt@kw{pt}
6836 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6837 \AddToHook{begindocument}{
6838   \ltx@ifpackageloaded{babel}{
6839     \makeatletter
6840     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6841     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6842       \input{problem-ngerman.ldf}
6843     }
6844     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6845       \input{problem-finnish.ldf}
6846     }
6847     \clist_if_in:NnT \l_tmpa_clist {french}{
6848       \input{problem-french.ldf}
6849     }
6850     \clist_if_in:NnT \l_tmpa_clist {russian}{
6851       \input{problem-russian.ldf}
6852     }
6853     \makeatother
6854   }{ }
6855 }
```

39.2 Problems and Solutions

We now prepare the KeyVal support for problems. The key macros just set appropriate internal macros.

```
6856 \keys_define:nn{ problem / problem }{
6857   id      .str_set_x:N = \l__problems_prob_id_str,
6858   pts     .tl_set:N   = \l__problems_prob_pts_tl,
6859   min     .tl_set:N   = \l__problems_prob_min_tl,
6860   title   .tl_set:N   = \l__problems_prob_title_tl,
6861   type    .tl_set:N   = \l__problems_prob_type_tl,
6862   refnum  .int_set:N   = \l__problems_prob_refnum_int
6863 }
6864 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6865 \str_clear:N \l__problems_prob_id_str
6866 \tl_clear:N \l__problems_prob_pts_tl
6867 \tl_clear:N \l__problems_prob_min_tl
6868 \tl_clear:N \l__problems_prob_title_tl
6869 \tl_clear:N \l__problems_prob_type_tl
6870 \int_zero_new:N \l__problems_prob_refnum_int
6871 \keys_set:nn { problem / problem }{ #1 }
6872 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6873   \let\l__problems_prob_refnum_int\undefined
6874 }
6875 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6876 \newcounter{problem}
6877 \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.

```

6878 \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

```

6879 \newcommand\prob@number{
6880   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6881     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6882   }{
6883     \int_if_exist:NTF \l__problems_prob_refnum_int {
6884       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6885     }{
6886       \prob@label\theproblem
6887     }
6888   }
6889 }

```

(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.

```

6890 \newcommand\prob@title[3]{%
6891   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6892     #2 \l__problems_inclprob_title_tl #3
6893   }{
6894     \tl_if_exist:NTF \l__problems_prob_title_tl {
6895       #2 \l__problems_prob_title_tl #3
6896     }{
6897       #1
6898     }
6899   }
6900 }

```

(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.

```

6901 \def\prob@heading{
6902   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6903   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6904 }

```

(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`

```

6905 \newenvironment{sproblem}[1][]{
6906   \__problems_prob_args:n{#1}%\sref@target%
6907   \@in@omtexttrue% we are in a statement (for inline definitions)
6908   \stepcounter{problem}\record@problem
6909   \def\current@section@level{\prob@problem@kw}
6910   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6911     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6912   }{
6913     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6914   }
6915   \str_if_exist:NTF \l__problems_inclprob_id_str {
6916     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6917   }{
6918     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6919   }
6920
6921
6922   \clist_set:No \l_tmpa_clist \sproblemtype
6923   \tl_clear:N \l_tmpa_tl
6924   \clist_map_inline:Nn \l_tmpa_clist {
6925     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6926       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6927     }
6928   }
6929   \tl_if_empty:NTF \l_tmpa_tl {
6930     \__problems_sproblem_start:
6931   }{
6932     \l_tmpa_tl
6933   }
6934   \stex_ref_new_doc_target:n \sproblemid
6935 }{
6936   \clist_set:No \l_tmpa_clist \sproblemtype
6937   \tl_clear:N \l_tmpa_tl
6938   \clist_map_inline:Nn \l_tmpa_clist {
6939     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
6940       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
6941     }

```



```

6942 }
6943 \tl_if_empty:NTF \l_tmpa_tl {
6944   \__problems_sproblem_end:
6945 }{
6946   \l_tmpa_tl
6947 }
6948
6949
6950 \smallskip
6951 }
6952
6953
6954 \cs_new_protected:Nn \__problems_sproblem_start: {
6955   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
6956 }
6957 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
6958
6959 \newcommand\stexpatchproblem[3][] {
6960   \str_set:Nx \l_tmpa_str{ #1 }
6961   \str_if_empty:NTF \l_tmpa_str {
6962     \tl_set:Nn \__problems_sproblem_start: { #2 }
6963     \tl_set:Nn \__problems_sproblem_end: { #3 }
6964   }{
6965     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
6966     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
6967   }
6968 }
6969
6970
6971 \bool_if:NT \c__problems_boxed_bool {
6972   \surroundwithmdframed{problem}
6973 }

```

\record@problem This macro records information about the problems in the *.aux file.

```

6974 \def\record@problem{
6975   \protected@write\@auxout{}
6976   {
6977     \string\@problem{\prob@number}
6978     {
6979       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
6980         \l__problems_inclprob_pts_tl
6981       }{
6982         \l__problems_prob_pts_tl
6983       }
6984     }%
6985     {
6986       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
6987         \l__problems_inclprob_min_tl
6988       }{
6989         \l__problems_prob_min_tl
6990       }
6991     }
6992   }
6993 }

```

(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).

```
6994 \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.

```
6995 \keys_define:nn { problem / solution }{
6996   id                .str_set_x:N = \l__problems_solution_id_str ,
6997   for               .tl_set:N   = \l__problems_solution_for_tl ,
6998   height           .dim_set:N   = \l__problems_solution_height_dim ,
6999   creators         .clist_set:N = \l__problems_solution_creators_clist ,
7000   contributors     .clist_set:N = \l__problems_solution_contributors_clist ,
7001   srccite          .tl_set:N    = \l__problems_solution_srccite_tl
7002 }
7003 \cs_new_protected:Nn \__problems_solution_args:n {
7004   \str_clear:N \l__problems_solution_id_str
7005   \tl_clear:N \l__problems_solution_for_tl
7006   \tl_clear:N \l__problems_solution_srccite_tl
7007   \clist_clear:N \l__problems_solution_creators_clist
7008   \clist_clear:N \l__problems_solution_contributors_clist
7009   \dim_zero:N \l__problems_solution_height_dim
7010   \keys_set:nn { problem / solution }{ #1 }
7011 }
```

the next step is to define a helper macro that does what is needed to start a solution.

```
7012 \newcommand\@startsolution[1][{}]{
7013   \__problems_solution_args:n { #1 }
7014   \@in@omtexttrue% we are in a statement.
7015   \bool_if:NF \c__problems_boxed_bool { \hrule }
7016   \smallskip\noindent
7017   {\textbf\prob@solution@kw : \enspace}
7018   \begin{small}
7019   \def\current@section@level{\prob@solution@kw}
7020   \ignorespacesandpars
7021 }
```

\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.

```
7022 \newcommand\startsolutions{
7023   \specialcomment{solution}{\@startsolution}{
7024     \bool_if:NF \c__problems_boxed_bool {
7025       \hrule\medskip
7026     }
7027     \end{small}%
7028   }
7029   \bool_if:NT \c__problems_boxed_bool {
7030     \surroundwithmdframed{solution}
7031   }
7032 }
```

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

\stopsolutions

```
7033 \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.

```
7034 \ifsolutions
7035   \startsolutions
7036 \else
7037   \stopsolutions
7038 \fi
```

exnote

```
7039 \bool_if:NTF \c__problems_notes_bool {
7040   \newenvironment{exnote}[1][]{
7041     \par\smallskip\hrule\smallskip
7042     \noindent\textbf{\prob@note@kw : }\small
7043   }{
7044     \smallskip\hrule
7045   }
7046 }{
7047   \excludecomment{exnote}
7048 }
```

hint

```
7049 \bool_if:NTF \c__problems_notes_bool {
7050   \newenvironment{hint}[1][]{
7051     \par\smallskip\hrule\smallskip
7052     \noindent\textbf{\prob@hint@kw :~ }\small
7053   }{
7054     \smallskip\hrule
7055   }
7056 \newenvironment{exhint}[1][]{
7057   \par\smallskip\hrule\smallskip
7058   \noindent\textbf{\prob@hint@kw :~ }\small
7059 }{
7060   \smallskip\hrule
7061 }
7062 }{
7063   \excludecomment{hint}
7064   \excludecomment{exhint}
7065 }
```

gnote

```
7066 \bool_if:NTF \c__problems_notes_bool {
7067   \newenvironment{gnote}[1][]{
7068     \par\smallskip\hrule\smallskip
7069     \noindent\textbf{\prob@gnote@kw : }\small
7070   }{
7071     \smallskip\hrule
7072   }
7073 }{
7074   \excludecomment{gnote}
7075 }
```

39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7076 \newenvironment{mcb}{
7077   \begin{enumerate}
7078 }{
7079   \end{enumerate}
7080 }

```

we define the keys for the mcc macro

```

7081 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7082   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7083     \bool_set_true:N #1
7084   }{
7085     \bool_set_false:N #1
7086   }
7087 }
7088 \keys_define:nn { problem / mcc }{
7089   id          .str_set_x:N = \l__problems_mcc_id_str ,
7090   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7091   T           .default:n   = { true } ,
7092   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7093   F           .default:n   = { true } ,
7094   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7095   Ttext       .code:n      = {
7096     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7097   } ,
7098   Ftext       .code:n      = {
7099     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7100   }
7101 }
7102 \cs_new_protected:Nn \l__problems_mcc_args:n {
7103   \str_clear:N \l__problems_mcc_id_str
7104   \tl_clear:N \l__problems_mcc_feedback_tl
7105   \bool_set_true:N \l__problems_mcc_t_bool
7106   \bool_set_true:N \l__problems_mcc_f_bool
7107   \bool_set_true:N \l__problems_mcc_Ttext_bool
7108   \bool_set_false:N \l__problems_mcc_Ftext_bool
7109   \keys_set:nn { problem / mcc }{ #1 }
7110 }

```

\mcc

```

7111 \newcommand\mcc[2][] {
7112   \l__problems_mcc_args:n{ #1 }
7113   \item #2
7114   \ifsolutions
7115     \
7116     \bool_if:NT \l__problems_mcc_t_bool {
7117       % TODO!
7118       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7119     }
7120     \bool_if:NT \l__problems_mcc_f_bool {

```

¹⁷EdNOTE: MK: maybe import something better here from a dedicated MC package

```

7121      % TODO!
7122      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7123    }
7124    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7125      !
7126    }{
7127      \l__problems_mcc_feedback_tl
7128    }
7129    \fi
7130  } %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.

```

7131
7132 \keys_define:nn{ problem / inclproblem }{
7133   id      .str_set:N = \l__problems_inclprob_id_str,
7134   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7135   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7136   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7137   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7138   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7139   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
7140 }
7141 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7142   \str_clear:N \l__problems_prob_id_str
7143   \tl_clear:N \l__problems_inclprob_pts_tl
7144   \tl_clear:N \l__problems_inclprob_min_tl
7145   \tl_clear:N \l__problems_inclprob_title_tl
7146   \tl_clear:N \l__problems_inclprob_type_tl
7147   \int_zero_new:N \l__problems_inclprob_refnum_int
7148   \str_clear:N \l__problems_inclprob_mhrepos_str
7149   \keys_set:nn { problem / inclproblem }{ #1 }
7150   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7151     \let\l__problems_inclprob_pts_tl\undefined
7152   }
7153   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7154     \let\l__problems_inclprob_min_tl\undefined
7155   }
7156   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7157     \let\l__problems_inclprob_title_tl\undefined
7158   }
7159   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7160     \let\l__problems_inclprob_type_tl\undefined
7161   }
7162   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7163     \let\l__problems_inclprob_refnum_int\undefined
7164   }
7165 }

```

```

7166
7167 \cs_new_protected:Nn \__problems_inclprob_clear: {
7168   \let\l__problems_inclprob_id_str\undefined
7169   \let\l__problems_inclprob_pts_tl\undefined
7170   \let\l__problems_inclprob_min_tl\undefined
7171   \let\l__problems_inclprob_title_tl\undefined
7172   \let\l__problems_inclprob_type_tl\undefined
7173   \let\l__problems_inclprob_refnum_int\undefined
7174   \let\l__problems_inclprob_mhrepos_str\undefined
7175 }
7176 \__problems_inclprob_clear:
7177
7178 \newcommand\includeproblem[2][ ]{
7179   \__problems_inclprob_args:n{ #1 }
7180   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7181     \input{#2}
7182   }{
7183     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7184       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7185     }
7186   }
7187   \__problems_inclprob_clear:
7188 }

```

(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.

```

7189 \AddToHook{enddocument}{
7190   \bool_if:NT \c__problems_pts_bool {
7191     \message{Total:~\arabic{pts}~points}
7192   }
7193   \bool_if:NT \c__problems_min_bool {
7194     \message{Total:~\arabic{min}~minutes}
7195   }
7196 }

```

The margin pars are reader-visible, so we need to translate

```

7197 \def\pts#1{
7198   \bool_if:NT \c__problems_pts_bool {
7199     \marginpar{#1~\prob@pt@kw}
7200   }
7201 }
7202 \def\min#1{
7203   \bool_if:NT \c__problems_min_bool {
7204     \marginpar{#1~\prob@min@kw}
7205   }
7206 }

```

`\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.

```

7207 \newcounter{pts}
7208 \def\show@pts{
7209   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7210     \bool_if:NT \c__problems_pts_bool {
7211       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7212       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7213     }
7214   }{
7215     \tl_if_exist:NT \l__problems_prob_pts_tl {
7216       \bool_if:NT \c__problems_pts_bool {
7217         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7218         \addtocounter{pts}{\l__problems_prob_pts_tl}
7219       }
7220     }
7221   }
7222 }

```

(End definition for `\show@pts`. This function is documented on page ??.)
and now the same for the minutes

`\show@min`

```

7223 \newcounter{min}
7224 \def\show@min{
7225   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7226     \bool_if:NT \c__problems_min_bool {
7227       \marginpar{\l__problems_inclprob_min_tl\ min}
7228       \addtocounter{min}{\l__problems_inclprob_min_tl}
7229     }
7230   }{
7231     \tl_if_exist:NT \l__problems_prob_min_tl {
7232       \bool_if:NT \c__problems_min_bool {
7233         \marginpar{\l__problems_prob_min_tl\ min}
7234         \addtocounter{min}{\l__problems_prob_min_tl}
7235       }
7236     }
7237   }
7238 }
7239 </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.

```
7240 \@@=hwexam>
7241 \*cls>
7242 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7243 \RequirePackage{l3keys2e}
7244 \DeclareOption*{
7245   \PassOptionsToClass{\CurrentOption}{document-structure}
7246   \PassOptionsToPackage{\CurrentOption}{stex}
7247   \PassOptionsToPackage{\CurrentOption}{hwexam}
7248   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7249 }
7250 \ProcessOptions
```

We load `omdoc.cls`, and the desired packages. For the L^AT_EXML bindings, we make sure the right packages are loaded.

```
7251 \LoadClass{document-structure}
7252 \RequirePackage{stex}
7253 \RequirePackage{hwexam}
7254 \RequirePackage{tikzinput}
7255 \RequirePackage{graphicx}
7256 \RequirePackage{a4wide}
7257 \RequirePackage{amssymb}
7258 \RequirePackage{amstext}
7259 \RequirePackage{amsmath}
```

Finally, we register another keyword for the `document` environment. We give a default assignment type to prevent errors


```

7260 \newcommand\assig@default@type{\hwexam@assignment@kw}
7261 \def\document@hwexamtype{\assig@default@type}
7262 <@@=document_structure>
7263 \keys_define:nn { document-structure / document }{
7264 id .str_set_x:N = \c_document_structure_document_id_str,
7265 hwexamtype .tl_set:N = \document@hwexamtype
7266 }
7267 <@@=hwexam>
7268 </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.

```
7269 \*package>
7270 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7271 \RequirePackage{13keys2e}
7272
7273 \newif\iftest\testfalse
7274 \DeclareOption{test}{\testtrue}
7275 \newif\ifmultiple\multiplefalse
7276 \DeclareOption{multiple}{\multipletrue}
7277 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7278 \ProcessOptions
```

Then we make sure that the necessary packages are loaded (in the right versions).

```
7279 \RequirePackage{keyval}[1997/11/10]
7280 \RequirePackage{problem}
```

`\hwexam@*@kw` For multilinguality, we define internal macros for keywords that can be specialized in `*.ldf` files.

```
7281 \newcommand\hwexam@assignment@kw{Assignment}
7282 \newcommand\hwexam@given@kw{Given}
7283 \newcommand\hwexam@due@kw{Due}
7284 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7285 blank~for~extra~space}
7286 \def\hwexam@minutes@kw{minutes}
7287 \newcommand\correction@probs@kw{prob.}
7288 \newcommand\correction@pts@kw{total}
7289 \newcommand\correction@reached@kw{reached}
7290 \newcommand\correction@sum@kw{Sum}
7291 \newcommand\correction@grade@kw{grade}
7292 \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

```

7293 \AddToHook{begindocument}{
7294 \ltx@ifpackageloaded{babel}{
7295 \makeatletter
7296 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7297 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7298 \input{hwexam-ngerman.ldf}
7299 }
7300 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7301 \input{hwexam-finnish.ldf}
7302 }
7303 \clist_if_in:NnT \l_tmpa_clist {french}{
7304 \input{hwexam-french.ldf}
7305 }
7306 \clist_if_in:NnT \l_tmpa_clist {russian}{
7307 \input{hwexam-russian.ldf}
7308 }
7309 \makeatother
7310 }{}
7311 }
7312

```

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.

```

7313 \newcounter{assignment}
7314 \numberproblemsin{assignment}
7315 \renewcommand\prob@label[1]{\assignment@number.#1}

```

We will prepare the keyval support for the `assignment` environment.

```

7316 \keys_define:nn { hwexam / assignment } {
7317 id .str_set:N = \l__hwexam_assign_id_str,
7318 number .int_set:N = \l__hwexam_assign_number_int,
7319 title .tl_set:N = \l__hwexam_assign_title_tl,
7320 type .tl_set:N = \l__hwexam_assign_type_tl,
7321 given .tl_set:N = \l__hwexam_assign_given_tl,
7322 due .tl_set:N = \l__hwexam_assign_due_tl,
7323 loadmodules .code:n = {
7324 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7325 }
7326 }
7327 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7328 \str_clear:N \l__hwexam_assign_id_str
7329 \int_set:Nn \l__hwexam_assign_number_int {-1}
7330 \tl_clear:N \l__hwexam_assign_title_tl
7331 \tl_clear:N \l__hwexam_assign_type_tl
7332 \tl_clear:N \l__hwexam_assign_given_tl
7333 \tl_clear:N \l__hwexam_assign_due_tl
7334 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7335 \keys_set:nn { hwexam / assignment }{ #1 }
7336 }

```

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.

```

7337 \newcommand\given@due[2]{
7338 \bool_lazy_all:nF {
7339 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7340 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7341 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7342 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7343 }{ #1 }
7344
7345 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7346 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7347 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7348 }
7349 }{
7350 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7351 }
7352
7353 \bool_lazy_or:nnF {
7354 \bool_lazy_and_p:nn {
7355 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7356 }{
7357 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7358 }
7359 }{
7360 \bool_lazy_and_p:nn {
7361 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7362 }{
7363 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7364 }
7365 }{ ,~ }
7366
7367 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7368 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7369 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7370 }
7371 }{
7372 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7373 }
7374
7375 \bool_lazy_all:nF {
7376 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7377 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7378 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7379 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7380 }{ #2 }
7381 }

```

`\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.

```

7382 \newcommand\assignment@title[3]{
7383 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7384 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7385 #1
7386 }{
7387 #2\l__hwexam_assign_title_tl#3
7388 }
7389 }{
7390 #2\l__hwexam_inclasssign_title_tl#3
7391 }
7392 }

```

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

`\assignment@number` Like `\assignment@title` only for the number, and no around part.

```

7393 \newcommand\assignment@number{
7394 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7395 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7396 \arabic{assignment}
7397 } {
7398 \int_use:N \l__hwexam_assign_number_int
7399 }
7400 }{
7401 \int_use:N \l__hwexam_inclasssign_number_int
7402 }
7403 }

```

(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.

```

7404 \newenvironment{assignment}[1][ ]{
7405 \__hwexam_assignment_args:n { #1 }
7406 %\sref@target
7407 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7408 \global\stepcounter{assignment}
7409 }{
7410 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7411 }
7412 \setcounter{problem}{0}
7413 \def\current@section@level{\document@hwexamtype}
7414 %\sref@label@id{\document@hwexamtype \thesection}
7415 \begin{@assignment}
7416 }{
7417 \end{@assignment}
7418 }

```

In the multi-assignment case we just use the omdoc environment for suitable sectioning.

```

7419 \def\ass@title{
7420 \protect\document@hwexamtype~\arabic{assignment}
7421 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7422 }
7423 \ifmultiple
7424 \newenvironment{@assignment}{
7425 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7426 \begin{sfragment}[loadmodules]{\ass@title}
7427 }{
7428 \begin{sfragment}{\ass@title}
7429 }
7430 }{
7431 \end{sfragment}
7432 }

```

for the single-page case we make a title block from the same components.

```

7433 \else
7434 \newenvironment{@assignment}{
7435 \begin{center}\bf
7436 \Large@title\strut\
7437 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7438 \large\given@due{--\}{}\{;\}{}
7439 \end{center}
7440 }{}
7441 \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.

```

7442 \keys_define:nn { hwexam / inclassignment } {
7443 %id .str_set_x:N = \l__hwexam_assign_id_str,
7444 number .int_set:N = \l__hwexam_inclassign_number_int,
7445 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7446 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7447 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7448 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7449 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7450 }
7451 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7452 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7453 \tl_clear:N \l__hwexam_inclassign_title_tl
7454 \tl_clear:N \l__hwexam_inclassign_type_tl
7455 \tl_clear:N \l__hwexam_inclassign_given_tl
7456 \tl_clear:N \l__hwexam_inclassign_due_tl
7457 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7458 \keys_set:nn { hwexam / inclassignment }{ #1 }
7459 }
7460 \__hwexam_inclassignment_args:n {}
7461
7462 \newcommand\inputassignment[2][ ]{

```

```

7463 \_hwexam_inclassnment_args:n { #1 }
7464 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7465   \input{#2}
7466 }{
7467   \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7468     \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7469   }
7470 }
7471 \_hwexam_inclassnment_args:n {}
7472 }
7473 \newcommand\includeassignment[2][]{
7474   \newpage
7475   \inputassignment[#1]{#2}
7476 }

```

(End definition for \in*assignment. This function is documented on page ??.)

41.4 Typesetting Exams

\quizheading

```

7477 \ExplSyntaxOff
7478 \newcommand\quizheading[1]{%
7479   \def\@tas{#1}%
7480   \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7481   \ifx\@tas\@empty\else%
7482     \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7483   \fi%
7484 }
7485 \ExplSyntaxOn

```

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

\testheading

```

7486
7487 \def\hwexamheader{\input{hwexam-default.header}}
7488
7489 \def\hwexamminutes{
7490   \tl_if_empty:NTF \testheading@duration {
7491     {\testheading@min}~\hwexam@minutes@kw
7492   }{
7493     \testheading@duration
7494   }
7495 }
7496
7497 \keys_define:nn { hwexam / testheading } {
7498   min .tl_set:N = \testheading@min,
7499   duration .tl_set:N = \testheading@duration,
7500   reqpts .tl_set:N = \testheading@reqpts,
7501   tools .tl_set:N = \testheading@tools
7502 }
7503 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7504   \tl_clear:N \testheading@min
7505   \tl_clear:N \testheading@duration

```

```

7506 \tl_clear:N \testheading@reqpts
7507 \tl_clear:N \testheading@tools
7508 \keys_set:nn { hwexam / testheading }{ #1 }
7509 }
7510 \newenvironment{testheading}[1][]{
7511 \_hwexam_testheading_args:n{ #1 }
7512 \newcount\check@time\check@time=\testheading@min
7513 \advance\check@time by -\theassignment@totalmin
7514 \newif\if@bonuspoints
7515 \tl_if_empty:NTF \testheading@reqpts {
7516 \@bonuspointsfalse
7517 }{
7518 \newcount\bonus@pts
7519 \bonus@pts=\theassignment@totalpts
7520 \advance\bonus@pts by -\testheading@reqpts
7521 \edef\bonus@pts{\the\bonus@pts}
7522 \@bonuspointstrue
7523 }
7524 \edef\check@time{\the\check@time}
7525
7526 \makeatletter\hwexamheader\makeatother
7527 }{
7528 \newpage
7529 }

```

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

\testspace

```

7530 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7531 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7532 \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.

```

7533 <@=problems>
7534 \renewcommand\@problem[3]{
7535 \stepcounter{assignment@probs}
7536 \def\__problemspts{#2}
7537 \ifx\__problemspts\@empty\else
7538 \addtocounter{assignment@totalpts}{#2}
7539 \fi
7540 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7541 \xdef\correction@probs{\correction@probs & #1}%
7542 \xdef\correction@pts{\correction@pts & #2}
7543 \xdef\correction@reached{\correction@reached &}

```



```

7544 }
7545 <@@=hwexam>

```

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

`\correction@table` This macro generates the correction table

```

7546 \newcounter{assignment@probs}
7547 \newcounter{assignment@totalpts}
7548 \newcounter{assignment@totalmin}
7549 \def\correction@probs{\correction@probs@kw}
7550 \def\correction@pts{\correction@pts@kw}
7551 \def\correction@reached{\correction@reached@kw}
7552 \stepcounter{assignment@probs}
7553 \newcommand\correction@table{
7554 \resizebox{\textwidth}{!}{%
7555 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7556 &\multicolumn{\theassignment@probs}{c|}|%|
7557 {\footnotesize\correction@forgrading@kw} &\\ \hline
7558 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7559 \correction@pts & \theassignment@totalpts & \\ \hline
7560 \correction@reached & & \[.7cm]\hline
7561 \end{tabular}}
7562 </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}

```