

The sTeX3 Package *

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

2022-03-19

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 2022-03-19)

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	103
25.1	Generic Path Handling	103
25.2	PWD and kpsewhich	105
25.3	File Hooks and Tracking	106
25.4	MathHub Repositories	107
25.5	Using Content in Archives	111
26	gTeX-References Implementation	116
26.1	Document URIs and URLs	116
26.2	Setting Reference Targets	118
26.3	Using References	120
27	gTeX-Modules Implementation	123
27.1	The smodule environment	127
27.2	Invoking modules	132
28	gTeX-Module Inheritance Implementation	134
28.1	SMS Mode	134
28.2	Inheritance	137

29	STEX-Symbols Implementation	142
29.1	Symbol Declarations	142
29.2	Notations	149
29.3	Variables	159
30	STEX-Terms Implementation	166
30.1	Symbol Invocations	166
30.2	Terms	173
30.3	Notation Components	177
30.4	Variables	180
30.5	Sequences	181
31	STEX-Structural Features Implementation	183
31.1	Imports with modification	184
31.2	The feature environment	191
31.3	Structure	192
32	STEX-Statements Implementation	202
32.1	Definitions	202
32.2	Assertions	207
32.3	Examples	211
32.4	Logical Paragraphs	213
33	The Implementation	219
33.1	Package Options	219
33.2	Proofs	219
33.3	Justifications	230
34	STEX-Others Implementation	232
35	STEX-Metatheory Implementation	233
36	Tikzinput Implementation	236
37	document-structure.sty Implementation	238
37.1	The document-structure Class	238
37.2	Class Options	238
37.3	Beefing up the document environment	239
37.4	Implementation: document-structure Package	239
37.5	Package Options	239
37.6	Document Structure	241
37.7	Front and Backmatter	244
37.8	Global Variables	246

38 NotesSlides – Implementation	247
38.1 Class and Package Options	247
38.2 Notes and Slides	249
38.3 Header and Footer Lines	253
38.4 Frame Images	254
38.5 Colors and Highlighting	255
38.6 Sectioning	256
38.7 Excursions	259
39 The Implementation	260
39.1 Package Options	260
39.2 Problems and Solutions	261
39.3 Multiple Choice Blocks	267
39.4 Including Problems	268
39.5 Reporting Metadata	269
40 Implementation: The hwexam Class	271
40.1 Class Options	271
41 Implementation: The hwexam Package	273
41.1 Package Options	273
41.2 Assignments	274
41.3 Including Assignments	277
41.4 Typesetting Exams	278
41.5 Leftovers	280

Part I

Manual



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



Boxes like this one explain how some $\text{\texttt{S}\TeX}$ 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,
- RuS_{TeX} 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, $\text{\texttt{\textit{S}T\textit{E}X}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

3.2 $\text{\texttt{\textit{S}T\textit{E}X}}$ Archives

3.2.1 The Local MathHub-Directory

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

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

1. If the $\text{\texttt{\textit{S}T\textit{E}X}}$ package is loaded with the option `mathhub=/path/to/mathhub`, then $\text{\texttt{\textit{S}T\textit{E}X}}$ will consider `/path/to/mathhub` as the local MathHub-directory.
2. If the `mathhub` package option is *not* set, but the macro `\mathhub` exists when the $\text{\texttt{\textit{S}T\textit{E}X}}$ -package is loaded, then this macro is assumed to point to the local MathHub-directory; i.e. `\def\mathhub{/path/to/mathhub}\usepackage{stex}` will set the MathHub-directory as `path/to/mathhub`.
3. Otherwise, $\text{\texttt{\textit{S}T\textit{E}X}}$ will attempt to retrieve the system variable `MATHHUB`, assuming it will point to the local MathHub-directory. Since this variant needs setting up only *once* and is machine-specific (rather than defined in tex code), it is compatible with collaborating and sharing tex content, and hence recommended.

3.2.2 The Structure of $\text{\texttt{\textit{S}T\textit{E}X}}$ Archives

An $\text{\texttt{\textit{S}T\textit{E}X}}$ 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 $\text{\texttt{\textit{S}T\textit{E}X}}$ system, it needs to be in `/user/foo/MathHub/smglom/calculus`.

Each such archive needs two subdirectories:

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

An additional `lib`-directory is optional, and is where $\text{\texttt{\textit{S}T\textit{E}X}}$ will look for files included via `\libinput`.

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

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

- `/source/mod/` – individual \TeX modules, containing symbol declarations, notations, and `\begin{paragraph}``[type=symdoc,for=...]` environments for “encyclopedic” symbol documentations
- `/source/def/` – definitions
- `/source/ex/` – examples
- `/source/thm/` – theorems, lemmata and proofs; preferably proofs in separate files to allow for multiple proofs for the same statement
- `/source/snip/` – individual text snippets such as remarks, explanations etc.
- `/source/frag/` – individual document fragments, ideally only `\inputref`ing snippets, definitions, examples etc. in some desirable order
- `/source/tikz/` – tikz images, as individual `.tex`-files
- `/source/pic/` – image files.

3.2.3 MANIFEST.MF-Files

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

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

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

- `id`: The name of the archive, including its group (e.g. `smglom/calculus`),
- `source-base` or
 - `ns`: The namespace from which all symbol and module URIs in this repository are formed, see (TODO),
- `narration-base`: The namespace from which all document URIs in this repository are formed, see (TODO),
- `url-base`: The URL that is formed as a basis for *external references*, see (TODO),
- `dependencies`: All archives that this archive depends on. \TeX ignores this field, but MMT can pick up on them to resolve dependencies, e.g. for `lmh install`.

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\hookrightarrow An \TeX module corresponds to an MMT/OMDOC *theory*. As such it
 \hookrightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \hookrightarrow `<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 `<module-URI>?<constant-name>`.

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

`\notation`

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

Example 5

Input:

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

Output:

First: a ; Second: b

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

`\comp`

Unfortunately, we have no highlighting whatsoever now. That is because we need to tell \TeX explicitly which parts of the notation are *notation components* which *should* be highlighted. We can do so with the `\comp` command.

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

Example 6

Input:

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

Output:

First: a ; Second: b



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

Note that it is required that

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

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

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



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

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

`\symdef`

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

Example 7

Input:

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

Output:

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

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

`\setnotation`

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

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

Operator Notations

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

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

Example 8

Input:

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

Output:

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

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

3.3.3 Argument Types

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

b-Type Arguments

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

\hookrightarrow **b-type** arguments behave exactly like **i-type** arguments within \TeX , but applications of binding operators, i.e. symbols with **b-type** arguments, are translated to $\text{\textbackslash OMBIND}$ -terms in $\text{\textbackslash OMDoc}$ / $\text{\textbackslash MMT}$, rather than $\text{\textbackslash OMA}$.

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

Example 9

Input:

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

Output:

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

where the variable x is now *bound* by the $\text{\textbackslash summation}$ -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. $\text{\textbackslash addition}\{a,b,c,d,e\}$ rather than having to write something like $\text{\textbackslash addition}\{a\}\{\text{\textbackslash addition}\{b\}\{\text{\textbackslash addition}\{c\}\{\text{\textbackslash addition}\{d\}\{e\}\}\}\}$!

$\text{\textbackslash notation}$ (and consequently $\text{\textbackslash symdef}$, 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. $\text{\textbackslash ascendingchain}\{S\}\{a,b,c,d,e\}\{t\}$ should yield $\forall a <_S b <_S c <_S d <_S e. t$. The "base"-notation for this operator is simply $\{\text{\textbackslash comp}\{\text{\textbackslash forall}\} \#2 \text{\textbackslash comp}\{.,\}\#3\}$, where $\#2$ represents the full notation fragment *accumulated* from $\{a,b,c,d,e\}$.

The *additional* argument to $\text{\textbackslash notation}$ (or $\text{\textbackslash symdef}$) 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 $\{\#1 \text{\textbackslash comp}\{<\}_{\#1} \#2\}$:

Example 10

Input:

```

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

```

Output:

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

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

Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

B-Type Arguments

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

Example 12

Input:

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

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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



More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

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

In the example above:



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

`\svar`

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

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

\vardef

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

Example 19

Input:

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

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

\varseq

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

This is best shown by example:

Example 20

Input:

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

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$$a_1 + \dots + a_n$$

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

Example 22

Input:

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

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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

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



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

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

The module `Foo` must either be declared in the



file $\langle\textit{top-directory}\rangle/\textit{some/path}/\textit{Foo}[\langle\textit{lang}\rangle].\textit{tex}$, or in $\langle\textit{top-directory}\rangle/\textit{some/path}[\langle\textit{lang}\rangle].\textit{tex}$ (which are checked in that order).

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

`\STEXexport`

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



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

A safer alternative is to use macro definition principles, that are safe to use even if the macro being defined already exists, and ideally are local to the current T_EX group, such as `\def` or `\let`.

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A **monoid** is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

`\instantiate`

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

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

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm: `mathstructure{<name>}` does in fact simply create a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
 $\hookrightarrow M$ `\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 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}{\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}}$
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 $\varMb!:=\mathstrut{\varMb{universe},\varMb{op}!,\varMb{unit}}{\varMb{op}!:\funtype{\varMb{universe},\varMb{universe}}{\varMb{universe}}}$
12 a \symname{monoid} on $\Int$...

```

Output:

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

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

3.4.4 The copymodule Environment

TODO: explain

Given modules:

Example 28

Input:

```

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

```

Output:

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

Example 29

Input:

```

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

```

Output:

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

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 30

Input:

```

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

```

Output:

3.5 Primitive Symbols (The \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 As expected, the above example is translated to OMDoc/MMT as an
 \hookrightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \hookrightarrow `<OMV name="m"/>` as arguments.

\arg

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

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

Example 34

Input:

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

Output:

Addition is...

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

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

Example 35

Input:

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

Output:

adding k yields...

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

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

Example 36

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

Chapter 5

sTeX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 37

Input:

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

Output:

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

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

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

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

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

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

Example 38

Input:

```

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

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

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

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

Now the **mathstructure monoid** contains two additional symbols, namely the axioms for associativity and that e is a unit. Note that both symbols do not represent the mere *propositions* that e.g. \circ is associative, but *the assertion that it is actually true* that \circ is associative.

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

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

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

5.2 Proofs

TODO

²Of course, $\text{\texttt{S}\text{\textsf{T}\text{\textsf{E}\text{\textsf{X}}}}$ can not check that the assertions are the “correct” ones – but if the assertions (both in **monoid** as well as those for addition and zero) are properly marked up, MMT can. **TODO: should**

Chapter 6

Highlighting and Presentation Customizations

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

The `stexthm` defines some default customizations that can be used, but of course many existing L^AT_EX templates come with their own `definition`, `theorem` and similar environments that authors are supposed (or even required) to use. Their concrete syntax however is usually not compatible with all the additional arguments that `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 `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>

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

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

<code>\stex_set_current_repository:n</code>

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

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

<code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code>
-------------------------------------	----------------------------------------------------------------------------

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

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\mhinput</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Both <code>\input</code> the file <code><filename></code> in archive <code><archive-ID></code> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html-mode</code> , inserting a <i>reference</i> to the file instead. Both also set <code>\ifinputref</code> to true.
<hr/> <hr/> <code>\addmhbibresource</code>	<code>\inputref[<archive-ID>]{<filename>}</code> Adds a <code>.bib</code> -file <code><filename></code> in archive <code><archive-ID></code> (relative to the top-directory of the archive!).
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf-</code> archive of the current archive group(s) (if existent) in descending order. Throws an error if no file by that name exists in any of the relevant <code>lib</code> -folders.
<hr/> <hr/> <code>\libusepackage</code>	<code>\libusepackage[<args>]{<filename>}</code> Like <code>\libinput</code> , but looks for <code>.sty</code> -files and calls <code>\usepackage[<meta{args}>]{<Arg{filename}>}</code> instead of <code>\input</code> . Throws an error, if none or more than one suitable package file is found.
<hr/> <hr/> <code>\mhgraphics</code> <code>\cmhgraphics</code>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrefpos</code> . It then resolves the file path in <code>\mhgraphics[mhrefpos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source-</code> folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 10

STEX-References

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

10.1 Macros and Environments

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

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 Using References

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

Chapter 11

STEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

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

`\stex_modules_current_namespace:`

Computes the current namespace as follows:

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

The result is stored in `\l_stex_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.

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

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

12.1.2 Imports and Inheritance

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

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

 $\backslash\text{stex_import_module_uri:nn}$

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

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

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

- **name**: An (OMDOC) name. By default equal to `<macroname>`.
- **type**: An (ideally semantic) term, representing a *type*. Not used by `STEX`, but passed on to MMT for semantic services.
- **def**: An (ideally semantic) term, representing a *definiens*. Not used by `STEX`, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:

- i** a “normal” argument, e.g. `\symdecl{plus}[args=ii]` allows for `\plus{2}{2}`.
- a** an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl{plus}[args=a]` allows for `\plus{2,2,2}`.
- b** a *variable* argument. Is treated by `STEX` like an *i*-argument, but an application is turned into an `OMBind` in OMDOC, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl{forall}[args=bi]` allows for `\forall{x \in \mathbb{N}}{x \geq 0}`.

<hr/> <hr/> <code>\stex_symdecl_do:n</code>	<p>Implements the core functionality of <code>\symdecl</code>, and is called by <code>\symdecl</code> and <code>\symdef</code>. Ultimately stores the symbol $\langle URI \rangle$ in the property list <code>\l_stex_symdecl_<URI>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>name</code> (string), • <code>module</code> (string), • <code>notations</code> (sequence of strings; initially empty), • <code>local</code> (boolean), • <code>type</code> (token list), • <code>args</code> (string of <code>is</code>, <code>as</code> and <code>bs</code>), • <code>arity</code> (integer string), • <code>assoc</code>s (integer string; number of associative arguments),
<hr/> <hr/> <code>\stex_all_symbols:n</code>	Iterates over all currently available symbols. Requires two <code>\seq_map_break:</code> to break fully.
<hr/> <hr/> <code>\stex_get_symbol:n</code>	Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...
<hr/> <code>\notation</code> <hr/>	$\text{\notation}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <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>	$\text{\stex_notation_do:nn}\{\langle URI \rangle\}\{\langle notations^+ \rangle\}$ <p>Implements the core functionality of <code>\notation</code>, and is called by <code>\notation</code> and <code>\symdef</code>. Ultimately stores the notation in the property list <code>\g_stex_notation_<URI>\#<variant>\#<lang>_prop</code> with fields:</p> <ul style="list-style-type: none"> • <code>symbol</code> (URI string), • <code>language</code> (string), • <code>variant</code> (string), • <code>opprec</code> (integer string), • <code>argprec</code>s (sequence of integer strings)
<hr/> <hr/> <code>\symdef</code> <hr/>	$\text{\symdef}[\langle args \rangle]\{\langle symbol \rangle\}\{\langle notations^+ \rangle\}$ <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>proofcomment</code> environment is much like a <code>step</code> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <code>\premise</code> .

17.2.5 Proof End Markers

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

\sproofend	The <code>sproof</code> package provides the <code>\sproofend</code> macro for this. If a different symbol for the proof end is to be used (e.g. <i>q.e.d</i>), then this can be obtained by specifying it using the
\sProofEndSymbol	<code>\sProofEndSymbol</code> configuration macro (e.g. by specifying <code>\sProofEndSymbol{q.e.d}</code>). Some of the proof structuring macros above will insert proof end symbols for subproofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set <code>proofend={}</code> in them or use use <code>\sProofEndSymbol{}</code> .

17.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

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

17.3 Limitations

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

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

Chapter 18

sTeX-Metatheory

The default meta theory for an sTeX module. Contains symbols so ubiquitous, that it is virtually impossible to describe any flexiformal content without them, or that are required to annotate even the most primitive symbols with meaningful (foundation-independent) “type”-annotations, or required for basic structuring principles (theorems, definitions).

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

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

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

Chapter 20

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

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

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

20.1 Introduction

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

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

DAG models of documents allow to replace the “Copy and Paste” in the source document with a label-and-reference model where document are shared in the document

source and the formatter does the copying during document formatting/presentation.⁴

20.2 The User Interface

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

20.2.1 Package and Class Options

The `document-structure` class accept the following options:

<code>class=<name></code>	load <code><name>.cls</code> instead of <code>article.cls</code>
<code>topsect=<sect></code>	The top-level sectioning level; the default for <code><sect></code> is <code>section</code>
<code>showignores</code>	show the the contents of the <code>ignore</code> environment after all
<code>showmeta</code>	show the metadata; see <code>metakeys.sty</code>
<code>showmods</code>	show modules; see <code>modules.sty</code>
<code>extrefs</code>	allow external references; see <code>sref.sty</code>
<code>defindex</code>	index definienda; see <code>statements.sty</code>
<code>minimal</code>	for testing; do not load any \LaTeX packages

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

20.2.2 Document Structure

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

```

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

```

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

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

`blindfragment`

\TeX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a `subsection*` as an introduction for a chapter). Therefore the `document-structure` package provides a variant `blindomgroup` that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

20.2.3 Ignoring Inputs

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

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

narrative/content markup In \TeX 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, \TeX 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[⟨URL⟩]{⟨label⟩}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `⟨URL⟩` that lets L^AT_EXML 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 L^AT_EX. 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 \TeX preamble of the course notes file. `\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` 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>
<code>notes</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 21.2.2). |
|-------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

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

21.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

⁵MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.

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

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

`\setsource`

`\setlicensing`

21.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add \TeX notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.⁷

`\frameimage`

`\mhframeimage`

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

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

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

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

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

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

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

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

21.2.5 Colors and Highlighting

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

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

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

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

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

```
\activateexcursion      where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions        call \inputref{<path>}. In this way, the3 \printexcursions macro (usually in the
                        appendix) will collect up all excursions that are specified in the main text.
```

Sometimes, we want to reference – in an excursion – part of another. We can use

```
\excursionref \excursionref{<label>} for that.
```

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

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

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

21.2.8 Miscellaneous

21.3 Limitations

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

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

Chapter 22

problem.sty: An Infrastructure for formatting Problems

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

22.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁶. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

22.2 The User Interface

22.2.1 Package Options

<code>solutions</code>	The <code>problem</code> package takes the options <code>solutions</code> (should solutions be output?), <code>notes</code>
<code>notes</code>	(should the problem notes be presented?), <code>hints</code> (do we give the hints?), <code>gnotes</code> (do we
<code>hints</code>	show grading notes?), <code>pts</code> (do we display the points awarded for solving the problem?),
<code>gnotes</code>	<code>min</code> (do we display the estimated minutes for problem soling). If theses are specified, then
<code>pts</code>	the corresponding auxiliary parts of the problems are output, otherwise, they remain
<code>min</code>	invisible.
<code>boxed</code>	The <code>boxed</code> option specifies that problems should be formatted in framed boxes so
<code>test</code>	that they are more visible in the text. Finally, the <code>test</code> option signifies that we are in
	a test situation, so this option does not show the solutions (of course), but leaves space
	for the students to solve them.
<code>mh</code>	The <code>mh</code> option turns on MathHub support; see [<code>Kohlhase:mss</code>].
<code>showmeta</code>	Finally, if the <code>showmeta</code> is set, then the metadata keys are shown (see [<code>Kohlhase:metakeys</code>]
	for details and customization options).

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

22.2.2 Problems and Solutions

problem The main environment provided by the **problem** package is (surprise surprise) the **problem** environment. It is used to mark up problems and exercises. The environment takes an optional KeyVal argument with the keys **id** as an identifier that can be reference later, **pts** for the points to be gained from this exercise in homework or quiz situations, **min** for the estimated minutes needed to solve the problem, and finally **title** for an informative title of the problem. For an example of a marked up problem see Figure 5 and the resulting markup see Figure 6.

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

Example 5: A marked up Problem

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

```
Problem 0.1 (Fitting Elephants)
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

hint The **hint** and **exnote** environments can be used in a **problem** environment to give hints and to make notes that elaborate certain aspects of the problem.

exnote

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

may arise in grading.

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

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

22.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

22.2.4 Including Problems

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

22.2.5 Reporting Metadata

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

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

22.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 23

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

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

Contents

23.1 Introduction

The `hwexam` package and class supplies an infrastructure that allows to format nice-looking assignment sheets by simply including problems from problem files marked up with the `problem` package [Kohlhase:problem]. It is designed to be compatible with `problems.sty`, and inherits some of the functionality.

23.2 The User Interface

23.2.1 Package and Class Options

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

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

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

23.2.2 Assignments

`assignment` This package supplies the `assignment` environment that groups problems into assignment sheets. It takes an optional KeyVal argument with the keys `number` (for the assignment number; if none is given, 1 is assumed as the default or — in multi-assignment documents — 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

sTeX -Basics Implementation

24.1 The sTeXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e,rustex}
6 \ProvidesExplClass{stex}{2022/03/03}{3.1.0}{sTeX document class}
7 \rustex_if:TF {
8   \LoadClass{article}
9 }{
10   \LoadClass[border=1px,varwidth]{standalone}
11   \setlength\textwidth{15cm}
12 }
13
14 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
15 \ProcessOptions
16
17 \RequirePackage{stex}
18 </cls>
```

24.2 Preliminaries

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

```

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

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

```

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

24.3 Messages and logging

```

42 <@@=stex_log>

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

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

```

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

Redirecting messages:

```

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

```

24.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATXML`:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`

`\latexml_if:TF`

```

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

```

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

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

```

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

```

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

`_stex_annotate_checkempty:n`

```

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

```

(End definition for `_stex_annotate_checkempty:n`.)

`\stex_if_do_html_p:`

Whether to (locally) produce HTML output

`\stex_if_do_html:TF`

```

103 \bool_new:N \_stex_html_do_output_bool
104 \bool_set_true:N \_stex_html_do_output_bool
105
106 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
107   \bool_if:nTF \_stex_html_do_output_bool
108     \prg_return_true: \prg_return_false:
109 }

```

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

`\stex_suppress_html:n`

Whether to (locally) produce HTML output

```

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

```

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

`\stex_annotate:nnx`

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

`\stex_annotate_invisible:n`

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

`\stex_annotate_invisible:nnn`

```

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

```

```

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

```

```

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

```

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

24.5 Babel Languages

```

215 <@@=stex_language>

```

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

```

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

```

```

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

```

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

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

```

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

```

24.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

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

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

`\ignorespacesandpars`

```
280 \protected\def\ignorespacesandpars{  
281   \begingroup\catcode13=10\relax  
282   \@ifnextchar\par{  
283     \endgroup\expandafter\ignorespacesandpars\@gobble  
284   }{  
285     \endgroup  
286   }  
287 }
```

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

`\MMTrule`

```
288 \NewDocumentCommand \MMTrule {m m}{  
289   \seq_set_split:Nnn \l_tmpa_seq , {#2}  
290   \int_zero:N \l_tmpa_int  
291   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{  
292     $\seq_map_inline:Nn \l_tmpa_seq {  
293       \int_incr:N \l_tmpa_int  
294       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}  
295     }$  
296   }  
297 }  
298  
299 \NewDocumentCommand \MMTinclude {m}{  
300   \stex_annotate_invisible:nnn{import}{#1}{}  
301 }  
302 \</package>
```

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

Chapter 25

STEX -MathHub Implementation

```
303 <*package>
304
305 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
306
307 <@@=stex_path>
308
309 Warnings and error messages
310 \msg_new:nnn{stex}{error/norepository}{
311   No~archive~#1~found~in~#2
312 }
313 \msg_new:nnn{stex}{error/notinarchive}{
314   Not~currently~in~an~archive,~but~\detokenize{#1}~
315   needs~one!
316 }
317 \msg_new:nnn{stex}{error/nofile}{
318   \detokenize{#1}~could~not~find~file~#2
319 }
320 \msg_new:nnn{stex}{error/twofiles}{
321   \detokenize{#1}~found~two~candidates~for~#2
322 }
```

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`

```
321 \cs_new_protected:Nn \stex_path_from_string:Nn {
322   \str_set:Nx \l_tmpa_str { #2 }
323   \str_if_empty:NTF \l_tmpa_str {
324     \seq_clear:N #1
325   }{
326     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
327     \sys_if_platform_windows:T{
328       \seq_clear:N \l_tmpa_tl
```

```

329     \seq_map_inline:Nn #1 {
330       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
331       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
332     }
333     \seq_set_eq:NN #1 \l_tmpa_tl
334   }
335   \stex_path_canonicalize:N #1
336 }
337 }
338

```

(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`

```

339 \cs_new_protected:Nn \stex_path_to_string:NN {
340   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
341 }
342
343 \cs_new:Nn \stex_path_to_string:N {
344   \seq_use:Nn #1 /
345 }

```

(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`

```

346 \str_const:Nn \c__stex_path_dot_str {.}
347 \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.

```

348 \cs_new_protected:Nn \stex_path_canonicalize:N {
349   \seq_if_empty:NF #1 {
350     \seq_clear:N \l_tmpa_seq
351     \seq_get_left:NN #1 \l_tmpa_tl
352     \str_if_empty:NT \l_tmpa_tl {
353       \seq_put_right:Nn \l_tmpa_seq {}
354     }
355     \seq_map_inline:Nn #1 {
356       \str_set:Nn \l_tmpa_tl { ##1 }
357       \str_if_eq:NNF \l_tmpa_tl \c__stex_path_dot_str {
358         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
359           \seq_if_empty:NNTF \l_tmpa_seq {
360             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
361               \c__stex_path_up_str
362             }
363           }{
364             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
365             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
366               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
367                 \c__stex_path_up_str
368               }
369             }{

```

```

370         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
371     }
372 }
373 }{
374     \str_if_empty:NF \l_tmpa_tl {
375         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
376     }
377 }
378 }
379 }
380 \seq_gset_eq:NN #1 \l_tmpa_seq
381 }
382 }

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

383 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
384     \seq_if_empty:NTF #1 {
385         \prg_return_false:
386     }{
387         \seq_get_left:NN #1 \l_tmpa_tl
388         \sys_if_platform_windows:TF{
389             \str_if_in:NnTF \l_tmpa_tl {:}{
390                 \prg_return_true:
391             }{
392                 \prg_return_false:
393             }
394         }{
395             \str_if_empty:NTF \l_tmpa_tl {
396                 \prg_return_true:
397             }{
398                 \prg_return_false:
399             }
400         }
401     }
402 }

```

(End definition for `\stex_path_if_absolute:N \underline{TF}` . This function is documented on page 48.)

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

403 \str_new:N\l_stex_kpsewhich_return_str
404 \cs_new_protected:Nn \stex_kpsewhich:n {
405     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
406     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
407     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
408 }

```

(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`

```

409 \sys_if_platform_windows:TF{
410   \begingroup\escapechar=-1\catcode'\=12
411   \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
412   \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
413   \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_
414   }}{
415     \stex_kpsewhich:n{-var-value~PWD}
416   }
417
418   \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
419   \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
420   \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

```

421 <@@=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

```

422 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

423 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
424 \stex_path_from_string:Nn \c_stex_mainfile_seq
425   \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`

```

426 \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`

```

427 \cs_new_protected:Nn \stex_filestack_push:n {
428   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
429   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
430     \stex_path_from_string:Nn\g_stex_currentfile_seq{
431       \c_stex_pwd_str/#1
432     }
433   }
434   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
435   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
436 }

```



```

478 }
479 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
480 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
481 }

```

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

```

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

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

498 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

`_stex_mathhub_find_manifest:N` Attempts to find the MANIFEST.MF in some file path and stores its path in `\l__stex_mathhub_manifest_file_seq`:

```

499 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
500   \seq_set_eq:NN\l_tmpa_seq #1
501   \bool_set_true:N\l_tmpa_bool
502   \bool_while_do:Nn \l_tmpa_bool {
503     \seq_if_empty:NTF \l_tmpa_seq {
504       \bool_set_false:N\l_tmpa_bool
505     } {
506       \file_if_exist:nTF{
507         \stex_path_to_string:N\l_tmpa_seq/MANIFEST.MF
508       } {
509         \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
510         \bool_set_false:N\l_tmpa_bool
511       } {
512         \file_if_exist:nTF{
513           \stex_path_to_string:N\l_tmpa_seq/META-INF/MANIFEST.MF
514         } {
515           \seq_put_right:Nn\l_tmpa_seq{META-INF}
516           \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}

```

```

517         \bool_set_false:N\l_tmpa_bool
518     }{
519         \file_if_exist:nTF{
520             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
521         }{
522             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
523             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
524             \bool_set_false:N\l_tmpa_bool
525         }{
526             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
527         }
528     }
529 }
530 }
531 }
532 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
533 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

534 \ior_new:N \c__stex_mathhub_manifest_ior

```

(End definition for \c__stex_mathhub_manifest_ior.)

__stex_mathhub_parse_manifest:n Stores the entries in manifest file in the corresponding property list:

```

535 \cs_new_protected:Nn \__stex_mathhub_parse_manifest:n {
536     \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
537     \ior_open:Nn \c__stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
538     \ior_map_inline:Nn \c__stex_mathhub_manifest_ior {
539         \str_set:Nn \l_tmpa_str {##1}
540         \exp_args:NNoo \seq_set_split:Nnn
541             \l_tmpb_seq \c_colon_str \l_tmpa_str
542         \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {
543             \exp_args:NNe \str_set:Nn \l_tmpb_tl {
544                 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
545             }
546             \exp_args:No \str_case:nnTF \l_tmpa_tl {
547                 {id} {
548                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
549                     { id } \l_tmpb_tl
550                 }
551                 {narration-base} {
552                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
553                     { narr } \l_tmpb_tl
554                 }
555                 {url-base} {
556                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
557                     { docurl } \l_tmpb_tl
558                 }
559                 {source-base} {
560                     \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
561                     { ns } \l_tmpb_tl
562                 }

```

```

563     {ns} {
564       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
565       { ns } \l_tmpb_tl
566     }
567     {dependencies} {
568       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
569       { deps } \l_tmpb_tl
570     }
571   }{}{}
572 }{}
573 }
574 \ior_close:N \c__stex_mathhub_manifest_ior
575 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

576 \cs_new_protected:Nn \stex_set_current_repository:n {
577   \stex_require_repository:n { #1 }
578   \prop_set_eq:Nc \l_stex_current_repository_prop {
579     c_stex_mathhub_#1_manifest_prop
580   }
581 }

```

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

`\stex_require_repository:n`

```

582 \cs_new_protected:Nn \stex_require_repository:n {
583   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
584     \stex_debug:nn{mathhub}{Opening~archive:~#1}
585     \_stex_mathhub_do_manifest:n { #1 }
586   }
587 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

588 %\prop_new:N \l_stex_current_repository_prop
589
590 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
591 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
592   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
593 } {
594   \_stex_mathhub_parse_manifest:n { main }
595   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
596   \l_tmpa_str
597   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
598   \c_stex_mathhub_main_manifest_prop
599   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
600   \stex_debug:nn{mathhub}{Current~repository:~
601     \prop_item:Nn \l_stex_current_repository_prop {id}
602   }
603 }

```

(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.

```

604 \cs_new_protected:Nn \stex_in_repository:nn {
605   \str_set:Nx \l_tmpa_str { #1 }
606   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
607   \str_if_empty:NTF \l_tmpa_str {
608     \prop_if_exist:NTF \l_stex_current_repository_prop {
609       \stex_debug:nn{mathhub}{do~in~current~repository:~\prop_item:Nn \l_stex_current_reposi
610       \exp_args:Ne \l_tmpa_cs{
611         \prop_item:Nn \l_stex_current_repository_prop { id }
612       }
613     }{
614       \l_tmpa_cs{}
615     }
616   }{
617     \stex_debug:nn{mathhub}{in~repository:~\l_tmpa_str}
618     \stex_require_repository:n \l_tmpa_str
619     \str_set:Nx \l_tmpa_str { #1 }
620     \exp_args:Nne \use:nn {
621       \stex_set_current_repository:n \l_tmpa_str
622       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
623     }{
624       \stex_debug:nn{mathhub}{switching~back~to:~
625       \prop_if_exist:NTF \l_stex_current_repository_prop {
626         \prop_item:Nn \l_stex_current_repository_prop { id }::~
627       \meaning\l_stex_current_repository_prop
628     }{
629       no~repository
630     }
631   }
632   \prop_if_exist:NTF \l_stex_current_repository_prop {
633     \stex_set_current_repository:n {
634       \prop_item:Nn \l_stex_current_repository_prop { id }
635     }
636   }{
637     \let\exp_not:N\l_stex_current_repository_prop\exp_not:N\undefined
638   }
639 }
640 }
641 }

```

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

25.5 Using Content in Archives

`\mhpath`

```

642 \def \mhpath #1 #2 {
643   \exp_args:Ne \tl_if_empty:nTF{#1}{
644     \c_stex_mathhub_str /
645     \prop_item:Nn \l_stex_current_repository_prop { id }
646     / source / #2
647   }{
648     \c_stex_mathhub_str / #1 / source / #2

```

```

649 }
650 }

```

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

`\inputref`
`\mhinput`

```

651 \newif \ifinputref \inputreffalse
652
653 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
654   \stex_in_repository:nn {#1} {
655     \ifinputref
656       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
657     \else
658       \inputreftrue
659       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
660     \inputreffalse
661   \fi
662 }
663 }
664 \NewDocumentCommand \mhinput { 0{} m}{
665   \stex_mhinput:nn{ #1 }{ #2 }
666 }
667
668 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
669   \stex_in_repository:nn {#1} {
670     \bool_lazy_any:nTF {
671       {\rustex_if_p:}
672       {\latexml_if_p:}
673     } {
674       \str_clear:N \l_tmpa_str
675       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
676         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
677       }
678       \stex_annotate_invisible:nnn{inputref}{
679         \l_tmpa_str / #2
680       }{}
681     }{
682       \begingroup
683         \inputreftrue
684         \tl_if_empty:nTF{ ##1 }{
685           \input{#2}
686         }{
687           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
688         }
689       \endgroup
690     }
691   }
692 }
693 \NewDocumentCommand \inputref { 0{} m}{
694   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
695 }

```

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

\addmhbibresource

```
696 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
697   \stex_in_repository:nn {#1} {
698     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
699   }
700 }
701 \newcommand\addmhbibresource[2][]{
702   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
703 }
```

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

\libinput

```
704 \cs_new_protected:Npn \libinput #1 {
705   \prop_if_exist:NF \l_stex_current_repository_prop {
706     \msg_error:nnn{stex}{error/notinarchive}\libinput
707   }
708   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
709     \msg_error:nnn{stex}{error/notinarchive}\libinput
710   }
711   \seq_clear:N \l__stex_mathhub_libinput_files_seq
712   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
713   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
714
715   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
716     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
717     \IfFileExists{ \l_tmpa_str }{
718       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
719     }{}
720     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
721     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
722   }
723
724   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
725   \IfFileExists{ \l_tmpa_str }{
726     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
727   }{}
728
729   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
730     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
731   }{
732     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
733       \input{ ##1 }
734     }
735   }
736 }
```

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

\libusepackage

```
737 \NewDocumentCommand \libusepackage {0{} m} {
738   \prop_if_exist:NF \l_stex_current_repository_prop {
739     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
740   }
```



```

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

```

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

`\mhgraphics`
`\cmhgraphics`

```

774
775 \AddToHook{begindocument}{
776   \ltx@ifpackageloaded{graphicx}{
777     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
778     \newcommand\mhgraphics[2][{}]{%
779       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
780       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
781       \newcommand\cmhgraphics[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
782     }{

```

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

`\lstinputmhlisting`
`\cmlstinputmhlisting`

```

783 \ltx@ifpackageloaded{listings}{
784   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
785   \newcommand\lstinputmhlisting[2][{}]{%
786     \def\lst@mhrepos{}\setkeys{lst}{#1}%
787     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}

```

```

788     \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
789   }{}
790 }
791
792 \end{package}

```

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

Chapter 26

STEX -References Implementation

```
793 <*package>
794
795 %%%%%%%%%%%%%% references.dtx %%%%%%%%%%%%%%
796
797 <@@=stex_refs>
    Warnings and error messages
798
```

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

```
799 %\iow_new:N \c__stex_refs_refs_iow
800 \AddToHook{begindocument}{
801 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
802 }
803 \AddToHook{enddocument}{
804 % \iow_close:N \c__stex_refs_refs_iow
805 }
```

`\STEXreftitle`

```
806 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
807
808 \NewDocumentCommand \STEXreftitle { m } {
809 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
810 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
811 \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:`

```
812 \cs_new_protected:Nn \stex_get_document_uri: {
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 { narr } \l_tmpa_str {
822       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
823     }
824   }
825
826   \str_if_empty:NTF \l_tmpa_str {
827     \str_set:Nx \l_stex_current_docns_str {
828       file:/\stex_path_to_string:N \l_tmpa_seq
829     }
830   }{
831     \bool_set_true:N \l_tmpa_bool
832     \bool_while_do:Nn \l_tmpa_bool {
833       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
834       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
835         {source} { \bool_set_false:N \l_tmpa_bool }
836       }{}{
837         \seq_if_empty:NT \l_tmpa_seq {
838           \bool_set_false:N \l_tmpa_bool
839         }
840       }
841     }
842
843     \seq_if_empty:NTF \l_tmpa_seq {
844       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
845     }{
846       \str_set:Nx \l_stex_current_docns_str {
847         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
848       }
849     }
850   }
851 }
```

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

`\l_stex_current_docurl_str`

```
852 \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:`

```
853 \cs_new_protected:Nn \stex_get_document_url: {
854   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
855   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
856   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

857 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
858 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
859
860 \str_clear:N \l_tmpa_str
861 \prop_if_exist:NT \l_stex_current_repository_prop {
862   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
863     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
864       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
865     }
866   }
867 }
868
869 \str_if_empty:NTF \l_tmpa_str {
870   \str_set:Nx \l_stex_current_docurl_str {
871     file:/\stex_path_to_string:N \l_tmpa_seq
872   }
873 }{
874   \bool_set_true:N \l_tmpa_bool
875   \bool_while_do:Nn \l_tmpa_bool {
876     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
877     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
878       {source} { \bool_set_false:N \l_tmpa_bool }
879     }{}{
880       \seq_if_empty:NT \l_tmpa_seq {
881         \bool_set_false:N \l_tmpa_bool
882       }
883     }
884   }
885
886   \seq_if_empty:NTF \l_tmpa_seq {
887     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
888   }{
889     \str_set:Nx \l_stex_current_docurl_str {
890       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
891     }
892   }
893 }
894 }

```

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

26.2 Setting Reference Targets

```

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

```

`\stex_ref_new_doc_target:n`

```

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

```

(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.

```

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

```

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

```

943 \AtEndDocument{
944   \def\stexauxadddocref#1 #2 {}{}
945 }

```

`\stex_ref_new_sym_target:n`

```

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

```

```

951     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
952   }
953 }{
954   \str_if_empty:NF \l__stex_refs_curr_label_str {
955     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
956     \immediate\write\@auxout{
957       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
958         \l__stex_refs_curr_label_str
959       }
960     }
961   }
962 }
963 }

```

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

26.3 Using References

```

964 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

965
966 \keys_define:nn { stex / sref } {
967   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
968   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
969   pre           .tl_set:N = \l__stex_refs_pre_tl ,
970   post          .tl_set:N = \l__stex_refs_post_tl ,
971 }
972 \cs_new_protected:Nn \__stex_refs_args:n {
973   \tl_clear:N \l__stex_refs_linktext_tl
974   \tl_clear:N \l__stex_refs_fallback_tl
975   \tl_clear:N \l__stex_refs_pre_tl
976   \tl_clear:N \l__stex_refs_post_tl
977   \str_clear:N \l__stex_refs_repo_str
978   \keys_set:nn { stex / sref } { #1 }
979 }

```

The actual macro:

```

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

```

```

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

```

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

`\srefsym`

```

1042 \NewDocumentCommand \srefsym { 0{} m}{
1043   \stex_get_symbol:n { #2 }
1044   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1045 }

```



```

1046
1047 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1048   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1049     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1050   }{
1051     \__stex_refs_args:n { #1 }
1052     \str_if_empty:NTF \l__stex_refs_indocument_str {
1053       \tl_if_exist:cTF{sref_sym_#2 _type}{
1054         % doc uri in \l_tmpb_str
1055         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1056         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1057           % reference
1058           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1059             \cs_if_exist:cTF{autoref}{
1060               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1061             }{
1062               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1063             }
1064           }{
1065             \ltx@ifpackageloaded{hyperref}{
1066               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1067             }{
1068               \l__stex_refs_linktext_tl
1069             }
1070           }
1071         }{
1072           % URL
1073           \ltx@ifpackageloaded{hyperref}{
1074             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1075           }{
1076             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1077           }
1078         }
1079       }{
1080         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1081       }
1082     }{
1083       % TODO
1084     }
1085   }
1086 }

```

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

\srefsymuri

```

1087 \cs_new_protected:Npn \srefsymuri #1 #2 {
1088   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1089 }

```

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

```

1090 </package>

```

Chapter 27

STEX -Modules Implementation

```
1091 <*package>
1092
1093 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1094
1095 <@@=stex_modules>
1096
1097   Warnings and error messages
1098   \msg_new:nnn{stex}{error/unknownmodule}{
1099     No~module~#1~found
1100   }
1101   \msg_new:nnn{stex}{error/syntax}{
1102     Syntax~error:~#1
1103   }
1104   \msg_new:nnn{stex}{error/siglanguage}{
1105     Module~#1~declares~signature~#2,~but~does~not~
1106     declare~its~language
1107   }
1108   \msg_new:nnn{stex}{warning/deprecated}{
1109     #1~is~deprecated;~please~use~#2~instead!
1110   }
1111   \msg_new:nnn{stex}{error/conflictingmodules}{
1112     Conflicting~imports~for~module~#1
1113   }
1114
1115 \l_stex_current_module_str The current module:
1116 \str_new:N \l_stex_current_module_str
1117
1118 (End definition for \l_stex_current_module_str. This variable is documented on page 54.)
1119
1120 \l_stex_all_modules_seq Stores all available modules
1121 \seq_new:N \l_stex_all_modules_seq
1122
1123 (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
1115 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1116   \str_if_empty:NTF \l_stex_current_module_str
1117   \prg_return_false: \prg_return_true:
1118 }

(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
1119 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1120   \prop_if_exist:cTF { c_stex_module_#1_prop }
1121   \prg_return_true: \prg_return_false:
1122 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1123 \cs_new_protected:Nn \stex_add_to_current_module:n {
1124   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1125 }
1126 \cs_new_protected:Npn \STEXexport {
1127   \begingroup
1128   \newlinechar=-1\relax
1129   \endlinechar=-1\relax
1130   %\catcode'\ = 9\relax
1131   \expandafter\endgroup\__stex_modules_export:n
1132 }
1133 \cs_new_protected:Nn \__stex_modules_export:n {
1134   \ignorespaces #1
1135   \stex_add_to_current_module:n { \ignorespaces #1 }
1136   \stex_smsmode_do:
1137 }
1138 \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
1139 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1142 }

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

```

```

\stex_add_import_to_current_module:n
1143 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1144   \str_set:Nx \l_tmpa_str { #1 }
1145   \exp_args:Nno
1146   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1147     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1148   }
1149 }

```

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

`\stex_collect_imports:n`

```

1150 \cs_new_protected:Nn \stex_collect_imports:n {
1151   \seq_clear:N \l_stex_collect_imports_seq
1152   \__stex_modules_collect_imports:n {#1}
1153 }
1154 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1155   \seq_map_inline:cn {c_stex_module_#1_imports} {
1156     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1157       \__stex_modules_collect_imports:n { ##1 }
1158     }
1159   }
1160   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1161     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1162   }
1163 }

```

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

`\stex_do_up_to_module:n`

```

1164 \int_new:N \l__stex_modules_group_depth_int
1165 \cs_new_protected:Nn \stex_do_up_to_module:n {
1166   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1167     #1
1168   }{
1169     #1
1170     \expandafter \tl_gset:Nn
1171     \csname l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1172     \expandafter\expandafter\expandafter\endcsname
1173     \expandafter\expandafter\expandafter { \csname
1174       l__stex_modules_aftergroup_\l_stex_current_module_str _tl\endcsname #1 }
1175     \aftergroup\__stex_modules_aftergroup_do:
1176   }
1177 }
1178 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1179   \stex_debug:nn{aftergroup}{\cs_meaning:c{
1180     l__stex_modules_aftergroup_\l_stex_current_module_str _tl
1181   }}
1182   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1183     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1184     \tl_gclear:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1185   }{
1186     \use:c{l__stex_modules_aftergroup_\l_stex_current_module_str _tl}
1187     \aftergroup\__stex_modules_aftergroup_do:
1188   }
1189 }
1190 \cs_new_protected:Nn \stex_reset_up_to_module:n {
1191   \expandafter\let\csname l__stex_modules_aftergroup_#1_tl\endcsname\undefined
1192 }

```

(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).

1193

(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.

```

1194 \str_new:N \l_stex_modules_ns_str
1195 \str_new:N \l_stex_modules_subpath_str
1196 \cs_new_protected:Nn __stex_modules_compute_namespace:nN {
1197   \str_set:Nx \l_tmpa_str { #1 }
1198   \seq_set_eq:NN \l_tmpa_seq #2
1199   % split off file extension
1200   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1201   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1202   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1203   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1204
1205   \bool_set_true:N \l_tmpa_bool
1206   \bool_while_do:Nn \l_tmpa_bool {
1207     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1208     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1209       {source} { \bool_set_false:N \l_tmpa_bool }
1210     }{}{
1211       \seq_if_empty:NT \l_tmpa_seq {
1212         \bool_set_false:N \l_tmpa_bool
1213       }
1214     }
1215   }
1216
1217   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1218   \str_if_empty:NTF \l_stex_modules_subpath_str {
1219     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1220   }{
1221     \str_set:Nx \l_stex_modules_ns_str {
1222       \l_tmpa_str/\l_stex_modules_subpath_str
1223     }
1224   }
1225 }
1226
1227 \cs_new_protected:Nn \stex_modules_current_namespace: {
1228   \str_clear:N \l_stex_modules_subpath_str
1229   \prop_if_exist:NTF \l_stex_current_repository_prop {
1230     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1231     __stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1232   }{
1233     % split off file extension
1234     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1235     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1236     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1237     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1238     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1239     \str_set:Nx \l_stex_modules_ns_str {

```

```

1240     file:/\stex_path_to_string:N \l_tmpa_seq
1241   }
1242 }
1243 }

```

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

27.1 The smodule environment

smodule arguments:

```

1244 \keys_define:nn { stex / module } {
1245   title      .tl_set:N      = \smodulename ,
1246   type       .str_set_x:N   = \smodulename ,
1247   id         .str_set_x:N   = \smoduleid ,
1248   deprecate  .str_set_x:N   = \l_stex_module_deprecate_str ,
1249   ns         .str_set_x:N   = \l_stex_module_ns_str ,
1250   lang       .str_set_x:N   = \l_stex_module_lang_str ,
1251   sig        .str_set_x:N   = \l_stex_module_sig_str ,
1252   creators   .str_set_x:N   = \l_stex_module_creators_str ,
1253   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1254   meta       .str_set_x:N   = \l_stex_module_meta_str ,
1255   srccite    .str_set_x:N   = \l_stex_module_srccite_str
1256 }
1257
1258 \cs_new_protected:Nn \__stex_modules_args:n {
1259   \str_clear:N \smodulename
1260   \str_clear:N \smodulename
1261   \str_clear:N \smoduleid
1262   \str_clear:N \l_stex_module_ns_str
1263   \str_clear:N \l_stex_module_deprecate_str
1264   \str_clear:N \l_stex_module_lang_str
1265   \str_clear:N \l_stex_module_sig_str
1266   \str_clear:N \l_stex_module_creators_str
1267   \str_clear:N \l_stex_module_contributors_str
1268   \str_clear:N \l_stex_module_meta_str
1269   \str_clear:N \l_stex_module_srccite_str
1270   \keys_set:nn { stex / module } { #1 }
1271 }
1272
1273 % module parameters here? In the body?
1274

```

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

```

1275 \cs_new_protected:Nn \stex_module_setup:nn {
1276   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1277   \str_set:Nx \l_stex_module_name_str { #2 }
1278   \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1279 \stex_if_in_module:TF {
1280   % Nested module
1281   \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}

```

```

1282     { ns } \l_stex_module_ns_str
1283     \str_set:Nx \l_stex_module_name_str {
1284       \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1285       { name } / \l_stex_module_name_str
1286   }
1287 }{
1288   % not nested:
1289   \str_if_empty:NT \l_stex_module_ns_str {
1290     \stex_modules_current_namespace:
1291     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1292     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1293       / {\l_stex_module_ns_str}
1294     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1295     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1296       \str_set:Nx \l_stex_module_ns_str {
1297         \stex_path_to_string:N \l_tmpa_seq
1298       }
1299     }
1300   }
1301 }

```

Next, we determine the language of the module:

```

1302   \str_if_empty:NT \l_stex_module_lang_str {
1303     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1304     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1305     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1306     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1307     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1308       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1309         inferred~from~file~name}
1310       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1311     }
1312   }
1313
1314   \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1315     \prop_get:NVNTF {c_stex_languages_prop} \l_stex_module_lang_str
1316     \l_tmpa_str {
1317       \ltx@ifpackageloaded{babel}{
1318         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1319       }{}
1320     } {
1321       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1322     }
1323   }}

```

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

```

1324   \str_if_empty:NTF \l_stex_module_sig_str {
1325     \exp_args:Nnx \prop_gset_from_keyval:cn {
1326       c_stex_module\_l_stex_module_ns_str?\l_stex_module_name_str _prop
1327     } {
1328       name      = \l_stex_module_name_str ,
1329       ns        = \l_stex_module_ns_str ,
1330       file      = \exp_not:o { \g_stex_currentfile_seq } ,

```

```

1331     lang      = \l_stex_module_lang_str ,
1332     sig       = \l_stex_module_sig_str ,
1333     deprecate = \l_stex_module_deprecate_str ,
1334     meta      = \l_stex_module_meta_str
1335 }
1336 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1337 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1338 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1339 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1340 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1341 \str_if_empty:NT \l_stex_module_meta_str {
1342   \str_set:Nx \l_stex_module_meta_str {
1343     \c_stex_metatheory_ns_str ? Metatheory
1344   }
1345 }
1346 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1347   \bool_set_true:N \l_stex_in_meta_bool
1348   \exp_args:Nx \stex_add_to_current_module:n {
1349     \bool_set_true:N \l_stex_in_meta_bool
1350     \stex_activate_module:n {\l_stex_module_meta_str}
1351     \bool_set_false:N \l_stex_in_meta_bool
1352   }
1353   \stex_activate_module:n {\l_stex_module_meta_str}
1354   \bool_set_false:N \l_stex_in_meta_bool
1355 }
1356 }{
1357   \str_if_empty:NT \l_stex_module_lang_str {
1358     \msg_error:nnxx{stex}{error/siglanguage}{
1359       \l_stex_module_ns_str?\l_stex_module_name_str
1360     }\l_stex_module_sig_str}
1361   }
1362
1363   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1364   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1365   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1366   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1367   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1368   \str_set:Nx \l_tmpa_str {
1369     \stex_path_to_string:N \l_tmpa_seq /
1370     \l_tmpa_str . \l_stex_module_sig_str .tex
1371   }
1372   \IfFileExists \l_tmpa_str {
1373     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1374       \str_clear:N \l_stex_current_module_str
1375       \seq_clear:N \l_stex_all_modules_seq
1376       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1377     }
1378   }{
1379     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1380   }
1381   \stex_if_smsmode:F {
1382     \stex_activate_module:n {

```



```

1383     \l_stex_module_ns_str ? \l_stex_module_name_str
1384   }
1385 }
1386 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1387 }
1388 \str_if_empty:NF \l_stex_module_deprecate_str {
1389   \msg_warning:nnxx{stex}{warning/deprecated}{
1390     Module~\l_stex_current_module_str
1391   }{
1392     \l_stex_module_deprecate_str
1393   }
1394 }
1395 \seq_put_right:Nx \l_stex_all_modules_seq {
1396   \l_stex_module_ns_str ? \l_stex_module_name_str
1397 }
1398 \tl_clear:c{l_stex_modules_aftergroup\l_stex_module_ns_str ? \l_stex_module_name_str _tl}
1399 }

```

(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}
1400 \cs_new_protected:Nn \__stex_modules_begin_module: {
1401   \stex_reactivate_macro:N \STEXexport
1402   \stex_reactivate_macro:N \importmodule
1403   \stex_reactivate_macro:N \symdecl
1404   \stex_reactivate_macro:N \notation
1405   \stex_reactivate_macro:N \symdef
1406 }
1407 \stex_debug:nn{modules}{
1408   New~module:\\
1409   Namespace:~\l_stex_module_ns_str\\
1410   Name:~\l_stex_module_name_str\\
1411   Language:~\l_stex_module_lang_str\\
1412   Signature:~\l_stex_module_sig_str\\
1413   Metatheory:~\l_stex_module_meta_str\\
1414   File:~\stex_path_to_string:N \g_stex_currentfile_seq
1415 }
1416
1417 \stex_if_smsmode:F{
1418   \begin{stex_annotate_env} {theory} {
1419     \l_stex_module_ns_str ? \l_stex_module_name_str
1420   }
1421
1422   \stex_annotate_invisible:nnn{header}{} {
1423     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1424     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1425     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1426       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1427     }
1428     \str_if_empty:NF \smodulotype {
1429       \stex_annotate:nnn{type}{\smodulotype}{}
1430     }

```

```

1431     }
1432   }
1433   % TODO: Inherit metatheory for nested modules?
1434 }
1435 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

(End definition for \_stex_modules_begin_module:.)

```

```

\_stex_modules_end_module: implements \end{module}

1436 \cs_new_protected:Nn \_stex_modules_end_module: {
1437   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}}
1438   \stex_reset_up_to_module:n \l_stex_current_module_str
1439 }

```

(End definition for _stex_modules_end_module:.)

The core environment

```

1440 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1441 \NewDocumentEnvironment { smodule } { 0{} m } {
1442   \stex_module_setup:nn{#1}{#2}
1443   \par
1444   \stex_if_smsmode:F{
1445     \tl_clear:N \l_tmpa_tl
1446     \clist_map_inline:Nn \smodulotype {
1447       \tl_if_exist:cT {\_stex_modules_smodule_##1_start:}{
1448         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_start:}}
1449       }
1450     }
1451     \tl_if_empty:NTF \l_tmpa_tl {
1452       \_stex_modules_smodule_start:
1453     }{
1454       \l_tmpa_tl
1455     }
1456   }
1457   \_stex_modules_begin_module:
1458   \str_if_empty:NF \smoduleid {
1459     \stex_ref_new_doc_target:n \smoduleid
1460   }
1461   \stex_smsmode_do:
1462 } {
1463   \_stex_modules_end_module:
1464   \stex_if_smsmode:F {
1465     \end{stex_annotate_env}
1466     \clist_set:No \l_tmpa_clist \smodulotype
1467     \tl_clear:N \l_tmpa_tl
1468     \clist_map_inline:Nn \l_tmpa_clist {
1469       \tl_if_exist:cT {\_stex_modules_smodule_##1_end:}{
1470         \tl_set:Nn \l_tmpa_tl {\use:c{\_stex_modules_smodule_##1_end:}}
1471       }
1472     }
1473     \tl_if_empty:NTF \l_tmpa_tl {
1474       \_stex_modules_smodule_end:
1475     }{
1476       \l_tmpa_tl
1477     }

```

```

1478 }
1479 }

\stexpatchmodule

1480 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1481 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1482
1483 \newcommand\stexpatchmodule[3] [] {
1484   \str_set:Nx \l_tmpa_str{ #1 }
1485   \str_if_empty:NTF \l_tmpa_str {
1486     \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1487     \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1488   }{
1489     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1490     \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1491   }
1492 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n

1493 \NewDocumentCommand \STEXModule { m } {
1494   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1495   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1496   \tl_set:Nn \l_tmpa_tl {
1497     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1498   }
1499   \seq_map_inline:Nn \l_stex_all_modules_seq {
1500     \str_set:Nn \l_tmpb_str { ##1 }
1501     \str_if_eq:eeT { \l_tmpa_str } {
1502       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1503     } {
1504       \seq_map_break:n {
1505         \tl_set:Nn \l_tmpa_tl {
1506           \stex_invoke_module:n { ##1 }
1507         }
1508       }
1509     }
1510   }
1511   \l_tmpa_tl
1512 }
1513
1514 \cs_new_protected:Nn \stex_invoke_module:n {
1515   \stex_debug:nn{modules}{Invoking~module~#1}
1516   \peek_charcode_remove:NTF ! {
1517     \__stex_modules_invoke_uri:nN { #1 }
1518   } {
1519     \peek_charcode_remove:NTF ? {
1520       \__stex_modules_invoke_symbol:nn { #1 }
1521     } {
1522       \msg_error:nnx{stex}{error/syntax}{

```

```

1523     ?~or~!~expected~after~
1524     \c_backslash_str STEXModule{#1}
1525   }
1526 }
1527 }
1528 }
1529
1530 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1531   \str_set:Nn #2 { #1 }
1532 }
1533
1534 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1535   \stex_invoke_symbol:n{#1?#2}
1536 }

```

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

\stex_activate_module:n

```

1537 \bool_new:N \l_stex_in_meta_bool
1538 \bool_set_false:N \l_stex_in_meta_bool
1539 \cs_new_protected:Nn \stex_activate_module:n {
1540   \stex_debug:nn{modules}{Activating~module~#1}
1541   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1542     \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1543   }
1544   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1545     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1546     \use:c{ c_stex_module_#1_code }
1547   }
1548 }

```

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

```

1549 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1550 <*package>
1551
1552 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1553
```

28.1 SMS Mode

```
1554 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1555 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1556 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1557 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1558
1559 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1560   \makeatletter
1561   \makeatother
1562   \ExplSyntaxOn
1563   \ExplSyntaxOff
1564   \rustexBREAK
1565 }
1566
1567 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1568   \symdef
1569   \importmodule
1570   \notation
1571   \symdecl
1572   \STEXexport
1573   \inlineass
1574   \inlinedef
1575   \inlineex
1576   \endinput
1577   \setnotation
```

```

1578 \copynotation
1579 \assign
1580 \renamedekl
1581 \donotcopy
1582 \instantiate
1583 }
1584
1585 \exp_args:Nn \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1586   \tl_to_str:n {
1587     smodule,
1588     copymodule,
1589     interpretmodule,
1590     sdefinition,
1591     sexample,
1592     sassertion,
1593     sparagraph,
1594     mathstructure
1595   }
1596 }

```

(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
1597 \bool_new:N \g__stex_smsmode_bool
1598 \bool_set_false:N \g__stex_smsmode_bool
1599 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1600   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1601 }

```

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

```

\_stex_smsmode_in_smsmode:nn
1602 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1603   \vbox_set:Nn \l_tmpa_box {
1604     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1605     \bool_gset_true:N \g__stex_smsmode_bool
1606     #2
1607     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1608   }
1609   \box_clear:N \l_tmpa_box
1610 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

```

\stex_file_in_smsmode:nn
1611 \quark_new:N \q__stex_smsmode_break
1612
1613 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1614   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1615   \stex_smsmode_do:
1616 }
1617
1618 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1619   \stex_filestack_push:n{#1}

```

```

1620 \seq_gclear:N \l__stex_smsmode_importmodules_seq
1621 % ----- new -----
1622 \__stex_smsmode_in_smsmode:nn{#1}{
1623   \let\importmodule\__stex_smsmode_importmodule:
1624   \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1625   \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1626   \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1627   \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1628   \everyeof{\q__stex_smsmode_break\noexpand}
1629   \expandafter\expandafter\expandafter
1630   \stex_smsmode_do:
1631   \csname @ @ input\endcsname "#1"\relax
1632 }
1633 % ----- new -----
1634 \__stex_smsmode_in_smsmode:nn{#1} {
1635   #2
1636   % ----- new -----
1637   \begingroup
1638   %\stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1639   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1640     \stex_import_module_uri:nn ##1
1641     \stex_import_require_module:nnnn
1642     \l_stex_import_ns_str
1643     \l_stex_import_archive_str
1644     \l_stex_import_path_str
1645     \l_stex_import_name_str
1646   }
1647   \endgroup
1648   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1649   % ----- new -----
1650   \everyeof{\q__stex_smsmode_break\noexpand}
1651   \expandafter\expandafter\expandafter
1652   \stex_smsmode_do:
1653   \csname @ @ input\endcsname "#1"\relax
1654 }
1655 \stex_filestack_pop:
1656 }

```

(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:

```

1657 \cs_new_protected:Npn \stex_smsmode_do: {
1658   \stex_if_smsmode:T {
1659     \__stex_smsmode_do:w
1660   }
1661 }
1662 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1663   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1664     \expandafter\if\expandafter\relax\noexpand#1
1665     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1666   }else\expandafter\__stex_smsmode_do:w\fi
1667 }{
1668   \__stex_smsmode_do:w % #1

```

```

1669 }
1670 }
1671 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1672   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1673     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1674       #1\__stex_smsmode_do:w
1675     }{
1676       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1677         #1
1678       }{
1679         \cs_if_eq:NNTF \begin #1 {
1680           \__stex_smsmode_check_begin:n
1681         }{
1682           \cs_if_eq:NNTF \end #1 {
1683             \__stex_smsmode_check_end:n
1684           }{
1685             \__stex_smsmode_do:w
1686           }
1687         }
1688       }
1689     }
1690   }
1691 }
1692
1693 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1694   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1695     \begin{#1}
1696   }{
1697     \__stex_smsmode_do:w
1698   }
1699 }
1700 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1701   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1702     \end{#1}\__stex_smsmode_do:w
1703   }{
1704     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1705   }
1706 }

```

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

28.2 Inheritance

```

1707 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1708 \cs_new_protected:Nn \stex_import_module_uri:nn {
1709   \str_set:Nx \l_stex_import_archive_str { #1 }
1710   \str_set:Nn \l_stex_import_path_str { #2 }
1711
1712   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1713   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1714   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1715 }

```



```

1716 \stex_modules_current_namespace:
1717 \bool_lazy_all:nTF {
1718   {\str_if_empty_p:N \l_stex_import_archive_str}
1719   {\str_if_empty_p:N \l_stex_import_path_str}
1720   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1721 }{
1722   \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1723   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1724 }{
1725   \str_if_empty:NT \l_stex_import_archive_str {
1726     \prop_if_exist:NT \l_stex_current_repository_prop {
1727       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1728     }
1729   }
1730   \str_if_empty:NTF \l_stex_import_archive_str {
1731     \str_if_empty:NF \l_stex_import_path_str {
1732       \str_set:Nx \l_stex_import_ns_str {
1733         \l_stex_module_ns_str / \l_stex_import_path_str
1734       }
1735     }
1736   }{
1737     \stex_require_repository:n \l_stex_import_archive_str
1738     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str\_manifest\_prop } { ns }
1739     \l_stex_import_ns_str
1740     \str_if_empty:NF \l_stex_import_path_str {
1741       \str_set:Nx \l_stex_import_ns_str {
1742         \l_stex_import_ns_str / \l_stex_import_path_str
1743       }
1744     }
1745   }
1746 }
1747 }

```

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

```

\l_stex_import_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str
1748 \str_new:N \l_stex_import_name_str
1749 \str_new:N \l_stex_import_archive_str
1750 \str_new:N \l_stex_import_path_str
1751 \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:nmmn {\{ns\}} {\{archive-ID\}} {\{path\}} {\{name\}}
1752 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1753   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1754
1755     %\stex_debug:nn{requiremodule}{Here:\~1:\~1\~2:\~2\~3:\~3\~4:\~4}
1756
1757     \exp_args:NNxx \seq_set_split:Nnn \l_tmpa_seq {\tl_to_str:n{/#}} {#4}
1758     \seq_get_left:NN \l_tmpa_seq \l_tmpc_str
1759
1760     %\stex_debug:nn{requiremodule}{Top-module:\l_tmpc_str}
1761

```

```

1762 % archive
1763 \str_set:Nx \l_tmpa_str { #2 }
1764 \str_if_empty:NTF \l_tmpa_str {
1765   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1766 } {
1767   \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1768   \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1769   \seq_put_right:Nn \l_tmpa_seq { source }
1770 }
1771
1772 % path
1773 \str_set:Nx \l_tmpb_str { #3 }
1774 \str_if_empty:NTF \l_tmpb_str {
1775   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / \l_tmpc_str }
1776
1777   \ltx@ifpackageloaded{babel} {
1778     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1779       { \language } \l_tmpb_str {
1780       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1781     }
1782   } {
1783     \str_clear:N \l_tmpb_str
1784   }
1785
1786   %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1787   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1788     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1789   }{
1790     %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1791     \IfFileExists{ \l_tmpa_str.tex }{
1792       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1793     }{
1794       % try english as default
1795       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1796       \IfFileExists{ \l_tmpa_str.en.tex }{
1797         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1798       }{
1799         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1800       }
1801     }
1802   }
1803
1804 } {
1805   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1806   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1807
1808   \ltx@ifpackageloaded{babel} {
1809     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1810       { \language } \l_tmpb_str {
1811       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1812     }
1813   } {
1814     \str_clear:N \l_tmpb_str
1815   }

```

```

1816
1817 \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1818
1819 %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex}
1820 \IfFileExists{ \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }{
1821   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.\l_tmpb_str.tex }
1822 }{
1823   %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.tex}
1824   \IfFileExists{ \l_tmpa_str/\l_tmpc_str.tex }{
1825     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.tex }
1826   }{
1827     % try english as default
1828     %\stex_debug:nn{modules}{Checking~\l_tmpa_str/\l_tmpc_str.en.tex}
1829     \IfFileExists{ \l_tmpa_str/\l_tmpc_str.en.tex }{
1830       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/\l_tmpc_str.en.tex }
1831     }{
1832       %\stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1833       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1834         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1835       }{
1836         %\stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1837         \IfFileExists{ \l_tmpa_str.tex }{
1838           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1839         }{
1840           % try english as default
1841           %\stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1842           \IfFileExists{ \l_tmpa_str.en.tex }{
1843             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1844           }{
1845             \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1846           }
1847         }
1848       }
1849     }
1850   }
1851 }
1852
1853
1854 \str_if_eq:eeF{\g__stex_importmodule_file_str}{\seq_use:Nn \g_stex_currentfile_seq /}{
1855   \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1856     \seq_clear:N \l_stex_all_modules_seq
1857     \str_clear:N \l_stex_current_module_str
1858     \str_set:Nx \l_tmpb_str { #2 }
1859     \str_if_empty:NF \l_tmpb_str {
1860       \stex_set_current_repository:n { #2 }
1861     }
1862     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1863   }
1864
1865   \stex_if_module_exists:nF { #1 ? #4 } {
1866     \msg_error:nnx{stex}{error/unknownmodule}{
1867       #1?#4~(in~file~\g__stex_importmodule_file_str)
1868     }
1869   }

```

```

1870     }
1871
1872   }
1873   \stex_activate_module:n { #1 ? #4 }
1874 }

```

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

`\importmodule`

```

1875 \NewDocumentCommand \importmodule { 0{} m } {
1876   \stex_import_module_uri:nn { #1 } { #2 }
1877   \stex_debug:nn{modules}{Importing~module:~
1878     \l_stex_import_ns_str ? \l_stex_import_name_str
1879   }
1880   \stex_import_require_module:nnnn
1881   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1882   { \l_stex_import_path_str } { \l_stex_import_name_str }
1883   \stex_if_smsmode:F {
1884     \stex_annotate_invisible:nnn
1885     {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1886   }
1887   \exp_args:Nx \stex_add_to_current_module:n {
1888     \stex_import_require_module:nnnn
1889     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1890     { \l_stex_import_path_str } { \l_stex_import_name_str }
1891   }
1892   \exp_args:Nx \stex_add_import_to_current_module:n {
1893     \l_stex_import_ns_str ? \l_stex_import_name_str
1894   }
1895   \stex_smsmode_do:
1896   \ignorespacesandpars
1897 }
1898 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1899 \NewDocumentCommand \usemodule { 0{} m } {
1900   \stex_if_smsmode:F {
1901     \stex_import_module_uri:nn { #1 } { #2 }
1902     \stex_import_require_module:nnnn
1903     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1904     { \l_stex_import_path_str } { \l_stex_import_name_str }
1905     \stex_annotate_invisible:nnn
1906     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1907   }
1908   \stex_smsmode_do:
1909   \ignorespacesandpars
1910 }

```

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

```

1911 \</package>

```

Chapter 29

STEX -Symbols Implementation

```
1912 <*package>
1913
1914 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1915
    Warnings and error messages
1916 \msg_new:nnn{stex}{error/wrongargs}{
1917   args~value~in~symbol~declaration~for~#1~
1918   needs~to~be~i,~a,~b~or~B,~but~#2~given
1919 }
1920 \msg_new:nnn{stex}{error/unknownsymbol}{
1921   No~symbol~#1~found!
1922 }
1923 \msg_new:nnn{stex}{error/seqlength}{
1924   Expected~#1~arguments;~got~#2!
1925 }
1926 \msg_new:nnn{stex}{error/unknownnotation}{
1927   Unknown~notation~#1~for~#2!
1928 }
```

29.1 Symbol Declarations

```
1929 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1930 \cs_new_protected:Nn \stex_all_symbols:n {
1931   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1932   \seq_map_inline:Nn \l_stex_all_modules_seq {
1933     \seq_map_inline:cn{c_stex_module_##1_constants}{
1934       \__stex_symdecl_all_symbols_cs{##1?####1}
1935     }
1936   }
1937 }
```

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

`\STEXsymbol`

```
1938 \NewDocumentCommand \STEXsymbol { m } {
1939   \stex_get_symbol:n { #1 }
1940   \exp_args:No
1941   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1942 }
```

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

`symdecl` arguments:

```
1943 \keys_define:nn { stex / symdecl } {
1944   name          .str_set_x:N = \l_stex_symdecl_name_str ,
1945   local         .bool_set:N = \l_stex_symdecl_local_bool ,
1946   args          .str_set_x:N = \l_stex_symdecl_args_str ,
1947   type          .tl_set:N = \l_stex_symdecl_type_tl ,
1948   deprecate     .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1949   align         .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1950   gfc           .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1951   specializes   .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1952   def           .tl_set:N = \l_stex_symdecl_definiens_tl ,
1953   assoc         .choices:nn =
1954     {bin,binl,binr,pre,conj,pwconj}
1955     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}}
1956 }
1957
1958 \bool_new:N \l_stex_symdecl_make_macro_bool
1959
1960 \cs_new_protected:Nn \__stex_symdecl_args:n {
1961   \str_clear:N \l_stex_symdecl_name_str
1962   \str_clear:N \l_stex_symdecl_args_str
1963   \str_clear:N \l_stex_symdecl_deprecate_str
1964   \str_clear:N \l_stex_symdecl_assoctype_str
1965   \bool_set_false:N \l_stex_symdecl_local_bool
1966   \tl_clear:N \l_stex_symdecl_type_tl
1967   \tl_clear:N \l_stex_symdecl_definiens_tl
1968
1969   \keys_set:nn { stex / symdecl } { #1 }
1970 }
```

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

```
1971
1972 \NewDocumentCommand \symdecl { s m O{} } {
1973   \__stex_symdecl_args:n { #3 }
1974   \IfBooleanTF #1 {
1975     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1976   } {
1977     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1978   }
1979   \stex_symdecl_do:n { #2 }
1980   \stex_smsmode_do:
1981 }
1982
1983 \cs_new_protected:Nn \stex_symdecl_do:nn {
```

```

1984 \__stex_symdecl_args:n{#1}
1985 \bool_set_false:N \l_stex_symdecl_make_macro_bool
1986 \stex_symdecl_do:n{#2}
1987 }
1988
1989 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1990 \cs_new_protected:Nn \stex_symdecl_do:n {
1991   \str_if_in_module:F {
1992     % TODO throw error? some default namespace?
1993   }
1994
1995   \str_if_empty:NT \l_stex_symdecl_name_str {
1996     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1997   }
1998
1999   \prop_if_exist:cT { l_stex_symdecl_
2000     \l_stex_current_module_str ?
2001     \l_stex_symdecl_name_str
2002   _prop
2003   }{
2004     % TODO throw error (beware of circular dependencies)
2005   }
2006
2007   \prop_clear:N \l_tmpa_prop
2008   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
2009   \seq_clear:N \l_tmpa_seq
2010   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
2011   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
2012
2013   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
2014     \str_if_empty:NF \l_stex_module_deprecate_str {
2015       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
2016     }
2017   }
2018   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
2019
2020   \exp_args:No \stex_add_constant_to_current_module:n {
2021     \l_stex_symdecl_name_str
2022   }
2023
2024   % arity/args
2025   \int_zero:N \l_tmpb_int
2026
2027   \bool_set_true:N \l_tmpa_bool
2028   \str_map_inline:Nn \l_stex_symdecl_args_str {
2029     \token_case_meaning:NnF ##1 {
2030       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2031       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2032       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2033       {\tl_to_str:n a} {

```

```

2034     \bool_set_false:N \l_tmpa_bool
2035     \int_incr:N \l_tmpb_int
2036   }
2037   {\tl_to_str:n B} {
2038     \bool_set_false:N \l_tmpa_bool
2039     \int_incr:N \l_tmpb_int
2040   }
2041   }{
2042     \msg_error:nnxx{stex}{error/wrongargs}{
2043       \l_stex_current_module_str ?
2044       \l_stex_symdecl_name_str
2045     }{##1}
2046   }
2047 }
2048 \bool_if:NTF \l_tmpa_bool {
2049   % possibly numeric
2050   \str_if_empty:NTF \l_stex_symdecl_args_str {
2051     \prop_put:Nnn \l_tmpa_prop { args } {}
2052     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2053   }{
2054     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2055     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2056     \str_clear:N \l_tmpa_str
2057     \int_step_inline:nn \l_tmpa_int {
2058       \str_put_right:Nn \l_tmpa_str i
2059     }
2060     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2061   }
2062 } {
2063   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2064   \prop_put:Nnx \l_tmpa_prop { arity }
2065     { \str_count:N \l_stex_symdecl_args_str }
2066 }
2067 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2068
2069 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2070   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2071 }{
2072   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2073 }
2074
2075 % semantic macro
2076
2077 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2078   \exp_args:Nx \stex_do_up_to_module:n {
2079     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2080       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2081     }}
2082   }
2083
2084   \bool_if:NF \l_stex_symdecl_local_bool {
2085     \exp_args:Nx \stex_add_to_current_module:n {
2086       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2087         \l_stex_current_module_str ? \l_stex_symdecl_name_str

```



```

2088     } }
2089   }
2090 }
2091 }
2092
2093 \stex_debug:nn{symbols}{New~symbol:~
2094   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2095   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2096   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2097   Definiens:~\exp_not:o {\l_stex_symdecl_definiens_tl}
2098 }
2099
2100 % circular dependencies require this:
2101
2102 \prop_if_exist:cF {
2103   \l_stex_symdecl_
2104   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2105   _prop
2106 } {
2107   \exp_args:Nx \stex_do_up_to_module:n {
2108     \prop_set_from_keyval:cn {
2109       \l_stex_symdecl_
2110       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2111       _prop
2112     } {\prop_to_keyval:N \l_tmpa_prop}
2113     \seq_clear:c {
2114       \l_stex_symdecl_
2115       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2116       _notations
2117     }
2118   }
2119 }
2120
2121 \bool_if:NF \l_stex_symdecl_local_bool {
2122   \exp_args:Nx
2123   \stex_add_to_current_module:n {
2124     \seq_clear:c {
2125       \l_stex_symdecl_
2126       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2127       _notations
2128     }
2129     \prop_set_from_keyval:cn {
2130       \l_stex_symdecl_
2131       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2132       _prop
2133     } {
2134       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2135       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2136       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2137       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2138       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2139       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2140       defined   = \prop_item:Nn \l_tmpa_prop { defined }   ,
2141     }
  
```

```

2142     }
2143   }
2144
2145   \stex_if_smsmode:F {
2146     % \exp_args:Nx \stex_do_up_to_module:n {
2147     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2148     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2149     %   }
2150     % }
2151   \stex_if_do_html:T {
2152     \stex_annotate_invisible:nnn {symdecl} {
2153       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2154     } {
2155       \tl_if_empty:NF \l_stex_symdecl_type_tl {
2156         \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2157       }
2158       \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_type_tl$} {
2159         \prop_item:Nn \l_tmpa_prop { args }
2160       }
2161       \stex_annotate_invisible:nnn{macroname}{\l_stex_symdecl_type_tl$} {
2162         \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2163           \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2164         }
2165       }
2166       \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2167         \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2168       }
2169     }
2170   }
2171 }
2172 }

```

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

`\stex_get_symbol:n`

```

2173 \str_new:N \l_stex_get_symbol_uri_str
2174
2175 \cs_new_protected:Nn \stex_get_symbol:n {
2176   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2177     \tl_set:Nn \l_tmpa_tl { #1 }
2178     \__stex_symdecl_get_symbol_from_cs:
2179   }{
2180     % argument is a string
2181     % is it a command name?
2182     \cs_if_exist:cTF { #1 }{
2183       \cs_set_eq:Nc \l_tmpa_tl { #1 }
2184       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2185       \str_if_empty:NNTF \l_tmpa_str {
2186         \exp_args:Nx \cs_if_eq:NNTF {
2187           \tl_head:N \l_tmpa_tl
2188         } \stex_invoke_symbol:n {
2189           \__stex_symdecl_get_symbol_from_cs:
2190         }{
2191           \__stex_symdecl_get_symbol_from_string:n { #1 }

```

```

2192     }
2193   } {
2194     \_stex_symdecl_get_symbol_from_string:n { #1 }
2195   }
2196   }{
2197     % argument is not a command name
2198     \_stex_symdecl_get_symbol_from_string:n { #1 }
2199     % \l_stex_all_symbols_seq
2200   }
2201 }
2202 \str_if_eq:eeF {
2203   \prop_item:cn {
2204     l_stex_symdecl\_l_stex_get_symbol_uri_str _prop
2205   }{ deprecate }
2206 }{ }{
2207   \msg_warning:nnxx{stex}{warning/deprecated}{
2208     Symbol~\l_stex_get_symbol_uri_str
2209   }{
2210     \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{ deprecate }
2211   }
2212 }
2213 }
2214
2215 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_string:n {
2216   \tl_set:Nn \l_tmpa_tl {
2217     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2218   }
2219   \str_set:Nn \l_tmpa_str { #1 }
2220   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2221
2222   \stex_all_symbols:n {
2223     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2224       \seq_map_break:n{ \seq_map_break:n{
2225         \tl_set:Nn \l_tmpa_tl {
2226           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2227         }
2228       }}
2229     }
2230   }
2231
2232   \l_tmpa_tl
2233 }
2234
2235 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2236   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2237   { \tl_tail:N \l_tmpa_tl }
2238   \tl_if_single:NTF \l_tmpa_tl {
2239     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2240       \exp_after:wN \str_set:Nn \exp_after:wN
2241       \l_stex_get_symbol_uri_str \l_tmpa_tl
2242     }{
2243       % TODO
2244       % tail is not a single group
2245     }

```

```

2246 }{
2247   % TODO
2248   % tail is not a single group
2249 }
2250 }

```

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

29.2 Notations

```

2251 <@@=stex_notation>
      notation arguments:
2252 \keys_define:nn { stex / notation } {
2253   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2254   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2255   prec .str_set_x:N = \l__stex_notation_prec_str ,
2256   op .tl_set:N = \l__stex_notation_op_tl ,
2257   primary .bool_set:N = \l__stex_notation_primary_bool ,
2258   primary .default:n = {true} ,
2259   unknown .code:n = \str_set:Nx
2260     \l__stex_notation_variant_str \l_keys_key_str
2261 }
2262
2263 \cs_new_protected:Nn \_stex_notation_args:n {
2264   \str_clear:N \l__stex_notation_lang_str
2265   \str_clear:N \l__stex_notation_variant_str
2266   \str_clear:N \l__stex_notation_prec_str
2267   \tl_clear:N \l__stex_notation_op_tl
2268   \bool_set_false:N \l__stex_notation_primary_bool
2269
2270   \keys_set:nn { stex / notation } { #1 }
2271 }

```

\notation

```

2272 \NewDocumentCommand \notation { s m O{}} {
2273   \_stex_notation_args:n { #3 }
2274   \tl_clear:N \l_stex_symdecl_definiens_tl
2275   \stex_get_symbol:n { #2 }
2276   \tl_set:Nn \l_stex_notation_after_do_tl {
2277     \__stex_notation_final:
2278     \IfBooleanTF#1{
2279       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2280     }{}
2281     \stex_smsmode_do:\ignorespacesandpars
2282   }
2283   \stex_notation_do:nnnnn
2284   { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2285   { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2286   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2287   { \l__stex_notation_prec_str }
2288 }
2289 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nnnnn

```

2290 \seq_new:N \l__stex_notation_precedences_seq
2291 \tl_new:N \l__stex_notation_opprec_tl
2292 \int_new:N \l__stex_notation_currarg_int
2293 \tl_new:N \stex_symbol_after_invokation_tl
2294
2295 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2296   \let\l_stex_current_symbol_str\relax
2297   \seq_clear:N \l__stex_notation_precedences_seq
2298   \tl_clear:N \l__stex_notation_opprec_tl
2299   \str_set:Nx \l__stex_notation_args_str { #1 }
2300   \str_set:Nx \l__stex_notation_arity_str { #2 }
2301   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2302   \str_set:Nx \l__stex_notation_prec_str { #4 }
2303
2304   % precedences
2305   \str_if_empty:NTF \l__stex_notation_prec_str {
2306     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2307       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2308     }{
2309       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2310     }
2311   } {
2312     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2313       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2314       \int_step_inline:nn { \l__stex_notation_arity_str } {
2315         \exp_args:NNo
2316         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2317       }
2318     }{
2319       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2320       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2321         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2322         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2323           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2324             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2325           \seq_map_inline:Nn \l_tmpa_seq {
2326             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2327           }
2328         }
2329       }{
2330         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2331           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2332         }{
2333           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2334         }
2335       }
2336     }
2337   }
2338
2339   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2340   \int_step_inline:nn { \l__stex_notation_arity_str } {
2341     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2342       \exp_args:NNo

```

```

2343     \seq_put_right:No \l__stex_notation_precedences_seq {
2344         \l__stex_notation_opprec_tl
2345     }
2346 }
2347 }
2348 \tl_clear:N \l_stex_notation_dummyargs_tl
2349
2350 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2351     \exp_args:NNe
2352     \cs_set:Npn \l_stex_notation_macrocode_cs {
2353         \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2354         { \l__stex_notation_suffix_str }
2355         { \l__stex_notation_opprec_tl }
2356         { \exp_not:n { #5 } }
2357     }
2358     \l_stex_notation_after_do_tl
2359 }{
2360     \str_if_in:NnTF \l__stex_notation_args_str b {
2361         \exp_args:Nne \use:nn
2362         {
2363             \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2364             \cs_set:Npn \l__stex_notation_arity_str } { {
2365                 \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2366                 { \l__stex_notation_suffix_str }
2367                 { \l__stex_notation_opprec_tl }
2368                 { \exp_not:n { #5 } }
2369             }}
2370     }{
2371         \str_if_in:NnTF \l__stex_notation_args_str B {
2372             \exp_args:Nne \use:nn
2373             {
2374                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2375                 \cs_set:Npn \l__stex_notation_arity_str } { {
2376                     \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2377                     { \l__stex_notation_suffix_str }
2378                     { \l__stex_notation_opprec_tl }
2379                     { \exp_not:n { #5 } }
2380                 } }
2381         }{
2382             \exp_args:Nne \use:nn
2383             {
2384                 \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2385                 \cs_set:Npn \l__stex_notation_arity_str } { {
2386                     \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2387                     { \l__stex_notation_suffix_str }
2388                     { \l__stex_notation_opprec_tl }
2389                     { \exp_not:n { #5 } }
2390                 } }
2391         }
2392     }
2393
2394     \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2395     \int_zero:N \l__stex_notation_currarg_int
2396     \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq

```

```

2397     \__stex_notation_arguments:
2398   }
2399 }

```

(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

```

2400 \cs_new_protected:Nn \__stex_notation_arguments: {
2401   \int_incr:N \l__stex_notation_currarg_int
2402   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2403     \l_stex_notation_after_do_tl
2404   }{
2405     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2406     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2407     \str_if_eq:NnTF \l_tmpa_str a {
2408       \__stex_notation_argument_assoc:nn{a}
2409     }{
2410       \str_if_eq:NnTF \l_tmpa_str B {
2411         \__stex_notation_argument_assoc:nn{B}
2412       }{
2413         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2414         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2415           { \_stex_term_math_arg:nnn
2416             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2417             { \l_tmpb_str }
2418             { ###\int_use:N \l__stex_notation_currarg_int }
2419           }
2420         }
2421         \__stex_notation_arguments:
2422       }
2423     }
2424   }
2425 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:nn

```

2426 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2427
2428   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2429     {\l__stex_notation_arity_str}{
2430       #2
2431     }
2432   \int_zero:N \l_tmpa_int
2433   \tl_clear:N \l_tmpa_tl
2434   \str_map_inline:Nn \l__stex_notation_args_str {
2435     \int_incr:N \l_tmpa_int
2436     \tl_put_right:Nx \l_tmpa_tl {
2437       \str_if_eq:nnTF {##1}{a}{ {} }{
2438         \str_if_eq:nnTF {##1}{B}{ {} }{
2439           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2440         }
2441       }
2442     }

```

```

2443 }
2444 \exp_after:wN\exp_after:wN\exp_after:wN \def
2445 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2446 \exp_after:wN\exp_after:wN\exp_after:wN ##
2447 \exp_after:wN\exp_after:wN\exp_after:wN 1
2448 \exp_after:wN\exp_after:wN\exp_after:wN ##
2449 \exp_after:wN\exp_after:wN\exp_after:wN 2
2450 \exp_after:wN\exp_after:wN\exp_after:wN {
2451   \exp_after:wN \exp_after:wN \exp_after:wN
2452   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2453     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2454   }
2455 }
2456
2457 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2458 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2459   \stex_term_math_assoc_arg:nnnn
2460   { #1\int_use:N \l__stex_notation_currarg_int }
2461   { \l_tmpa_str }
2462   { ####\int_use:N \l__stex_notation_currarg_int }
2463   { \l_tmpa_cs {####1} {####2} }
2464 } }
2465 \__stex_notation_arguments:
2466 }

```

(End definition for _stex_notation_argument_assoc:nn.)

_stex_notation_final: Called after processing all notation arguments

```

2467 \cs_new_protected:Nn \__stex_notation_final: {
2468 % \exp_args:Nne \use:nn
2469 % {
2470 % \cs_generate_from_arg_count:cNnn {
2471 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2472 %   \l__stex_notation_suffix_str
2473 %   _cs
2474 % }
2475 % \cs_set:Npn \l__stex_notation_arity_str } { {
2476 %   \exp_after:wN \exp_after:wN \exp_after:wN
2477 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2478 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
2479 % } }
2480
2481 % \tl_if_empty:NF \l__stex_notation_op_tl {
2482 %   \cs_set:cpx {
2483 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2484 %     \l__stex_notation_suffix_str
2485 %     _cs
2486 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2487 % }
2488
2489 \exp_args:Nx \stex_do_up_to_module:n {
2490   \cs_generate_from_arg_count:cNnn {
2491     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2492     \l__stex_notation_suffix_str

```



```

2493   _cs
2494 } \cs_set:Npn {\l__stex_notation_arity_str} {
2495   \exp_after:wN \exp_after:wN \exp_after:wN
2496   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2497   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2498 }
2499 \tl_if_empty:NF \l__stex_notation_op_tl {
2500   \cs_set:cpn {
2501     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2502     \l__stex_notation_suffix_str
2503     _cs
2504   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2505 }
2506 }
2507
2508 \exp_args:Ne
2509 \stex_add_to_current_module:n {
2510   \cs_generate_from_arg_count:cNnn {
2511     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2512     \l__stex_notation_suffix_str
2513     _cs
2514   } \cs_set:Npn {\l__stex_notation_arity_str} {
2515     \exp_after:wN \exp_after:wN \exp_after:wN
2516     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2517     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2518   }
2519   \tl_if_empty:NF \l__stex_notation_op_tl {
2520     \cs_set:cpn {
2521       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2522       \l__stex_notation_suffix_str
2523       _cs
2524     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2525   }
2526 }
2527
2528 \stex_debug:nn{symbols}{
2529   Notation~\l__stex_notation_suffix_str
2530   ~for~\l_stex_get_symbol_uri_str^^J
2531   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2532   Argument~precedences:~
2533   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2534   Notation: \cs_meaning:c {
2535     stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2536     \l__stex_notation_suffix_str
2537     _cs
2538   }
2539 }
2540
2541 \exp_args:Ne
2542 \stex_do_up_to_module:n {
2543   \exp_not:N \seq_if_exist:cT { l_stex_symdecl_\l_stex_get_symbol_uri_str _notations }{
2544     \seq_put_right:cn {
2545       l_stex_symdecl_\l_stex_get_symbol_uri_str
2546       _notations

```

```

2547     } {\l__stex_notation_suffix_str}
2548   }
2549 }
2550 \exp_args:Ne
2551 \stex_add_to_current_module:n {
2552   \seq_put_right:cn {
2553     l_stex_symdecl_l\l_stex_get_symbol_uri_str
2554     _notations
2555   } { \l__stex_notation_suffix_str }
2556 }
2557
2558 \stex_if_smsmode:F {
2559
2560   % HTML annotations
2561   \stex_if_do_html:T {
2562     \stex_annotate_invisible:nnn { notation }
2563     { \l_stex_get_symbol_uri_str } {
2564       \stex_annotate_invisible:nnn { notationfragment }
2565       { \l__stex_notation_suffix_str }{}
2566       \stex_annotate_invisible:nnn { precedence }
2567       { \l__stex_notation_prec_str }{}
2568
2569       \int_zero:N \l_tmpa_int
2570       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2571       \tl_clear:N \l_tmpa_tl
2572       \int_step_inline:nn { \l__stex_notation_arity_str }{
2573         \int_incr:N \l_tmpa_int
2574         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2575         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2576         \str_if_eq:VnTF \l_tmpb_str a {
2577           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2578             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2579             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2580           } }
2581         }{
2582           \str_if_eq:VnTF \l_tmpb_str B {
2583             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2584               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2585               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2586             } }
2587           }{
2588             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2589               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2590             } }
2591           }
2592         }
2593       }
2594       \stex_annotate_invisible:nnn { notationcomp }{}{
2595         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2596         $ \exp_args:Nno \use:nn { \use:c {
2597           stex_notation_ \l_stex_current_symbol_str
2598           \c_hash_str \l__stex_notation_suffix_str _cs
2599         } } { \l_tmpa_tl } $
2600       }

```

```

2601     }
2602   }
2603 }
2604 }

```

(End definition for `_stex_notation_final:`.)

`\setnotation`

```

2605 \keys_define:nn { stex / setnotation } {
2606   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2607   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2608   unknown .code:n = \str_set:Nx
2609     \l__stex_notation_variant_str \l_keys_key_str
2610 }
2611
2612 \cs_new_protected:Nn \stex_setnotation_args:n {
2613   \str_clear:N \l__stex_notation_lang_str
2614   \str_clear:N \l__stex_notation_variant_str
2615   \keys_set:nn { stex / setnotation } { #1 }
2616 }
2617
2618 \cs_new_protected:Nn \stex_setnotation:n {
2619   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2620     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2621     \exp_args:Nx \stex_do_up_to_module:n {
2622       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2623         \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2624         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2625         \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2626         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2627       }
2628     }
2629   \exp_args:Nx \stex_add_to_current_module:n {
2630     \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2631     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2632     \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2633     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2634   }
2635   \stex_debug:nn {notations}{
2636     Setting~default~notation~
2637     {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2638     #1 \\
2639     \expandafter\meaning\csname
2640     l_stex_symdecl_#1 _notations\endcsname
2641   }
2642   }{
2643     \msg_error:nxxx{stex}{unknownnotation}{\l__stex_notation_variant_str \c_hash_str \l__s
2644   }
2645 }
2646
2647 \NewDocumentCommand \setnotation {m m} {
2648   \stex_get_symbol:n { #1 }
2649   \stex_setnotation_args:n { #2 }
2650   \stex_setnotation:n{\l_stex_get_symbol_uri_str}

```

```

2651 \stex_smsmode_do:\ignorespacesandpars
2652 }
2653
2654 \cs_new_protected:Nn \stex_copy_notations:nn {
2655   \stex_debug:nn {notations}{
2656     Copying~notations~from~#2~to~#1\\
2657     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2658   }
2659   \tl_clear:N \l_tmpa_tl
2660   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2661     \tl_put_right:Nn \l_tmpa_tl { {##} ##1} }
2662   }
2663   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2664     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2665     \edef \l_tmpa_tl {
2666       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2667       \exp_after:wN\exp_after:wN\exp_after:wN {
2668         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2669       }
2670     }
2671     \exp_args:Nx
2672     \stex_add_to_current_module:n {
2673       \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2674         \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2675         \cs_generate_from_arg_count:cNnn {
2676           stex_notation_ #1 \c_hash_str ##1 _cs
2677         } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2678           \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2679         }
2680         \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2681           \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2682             {\exp_args:NNo\exp_args:No\exp_not:n{\csgname stex_op_notation_ #2\c_hash_str ##1
2683             }
2684           }
2685         }
2686         \exp_args:Nx
2687         \stex_do_up_to_module:n {
2688           \exp_not:N \seq_if_exist:cT{l_stex_symdecl_#1_notations}{
2689             \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2690             \cs_generate_from_arg_count:cNnn {
2691               stex_notation_ #1 \c_hash_str ##1 _cs
2692             } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2693               \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2694             }
2695             \cs_if_exist:cT{stex_op_notation_ #2\c_hash_str ##1 _cs}{
2696               \tl_set:cn{stex_op_notation_ #1\c_hash_str ##1 _cs}
2697                 {\exp_args:NNo\exp_args:No\exp_not:n{\csgname stex_op_notation_ #2\c_hash_str ##1
2698                 }
2699               }
2700           }
2701         }
2702       }
2703     }
2704     \NewDocumentCommand \copynotation {m m} {

```

```

2705 \stex_get_symbol:n { #1 }
2706 \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2707 \stex_get_symbol:n { #2 }
2708 \exp_args:Noo
2709 \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2710 \exp_args:Nx \stex_add_to_current_module:n{
2711   \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2712 }
2713 \stex_smsmode_do:\ignorespacesandpars
2714 }
2715

```

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

\symdef

```

2716 \keys_define:nn { stex / symdef } {
2717   name .str_set_x:N = \l_stex_symdecl_name_str ,
2718   local .bool_set:N = \l_stex_symdecl_local_bool ,
2719   args .str_set_x:N = \l_stex_symdecl_args_str ,
2720   type .tl_set:N = \l_stex_symdecl_type_tl ,
2721   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2722   op .tl_set:N = \l__stex_notation_op_tl ,
2723   lang .str_set_x:N = \l__stex_notation_lang_str ,
2724   variant .str_set_x:N = \l__stex_notation_variant_str ,
2725   prec .str_set_x:N = \l__stex_notation_prec_str ,
2726   assoc .choices:nn =
2727     {bin,binl,binr,pre,conj,pwconj}
2728     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},
2729   unknown .code:n = \str_set:Nx
2730     \l__stex_notation_variant_str \l_keys_key_str
2731 }
2732
2733 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2734   \str_clear:N \l_stex_symdecl_name_str
2735   \str_clear:N \l_stex_symdecl_args_str
2736   \str_clear:N \l_stex_symdecl_assoctype_str
2737   \bool_set_false:N \l_stex_symdecl_local_bool
2738   \tl_clear:N \l_stex_symdecl_type_tl
2739   \tl_clear:N \l_stex_symdecl_definiens_tl
2740   \str_clear:N \l__stex_notation_lang_str
2741   \str_clear:N \l__stex_notation_variant_str
2742   \str_clear:N \l__stex_notation_prec_str
2743   \tl_clear:N \l__stex_notation_op_tl
2744
2745   \keys_set:nn { stex / symdef } { #1 }
2746 }
2747
2748 \NewDocumentCommand \symdef { m O{} } {
2749   \__stex_notation_symdef_args:n { #2 }
2750   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2751   \stex_symdecl_do:n { #1 }
2752   \tl_set:Nn \l_stex_notation_after_do_tl {
2753     \__stex_notation_final:
2754     \stex_smsmode_do:\ignorespacesandpars

```

```

2755 }
2756 \str_set:Nx \l_stex_get_symbol_uri_str {
2757   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2758 }
2759 \exp_args:Nx \stex_notation_do:nnnnn
2760 { \prop_item:cn {l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { args } }
2761 { \prop_item:cn { l_stex_symdecl_\l_stex_get_symbol_uri_str _prop } { arity } }
2762 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2763 { \l__stex_notation_prec_str}
2764 }
2765 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

29.3 Variables

```

2766 <@@=stex_variables>
2767
2768 \keys_define:nn { stex / vardef } {
2769   name .str_set_x:N = \l__stex_variables_name_str ,
2770   args .str_set_x:N = \l__stex_variables_args_str ,
2771   type .tl_set:N    = \l__stex_variables_type_tl ,
2772   def  .tl_set:N    = \l__stex_variables_def_tl ,
2773   op   .tl_set:N    = \l__stex_variables_op_tl ,
2774   prec .str_set_x:N = \l__stex_variables_prec_str ,
2775   assoc .choices:nn =
2776     {bin,binl,binr,pre,conj,pwconj}
2777     {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2778   bind .choices:nn =
2779     {forall,exists}
2780     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2781 }
2782
2783 \cs_new_protected:Nn \__stex_variables_args:n {
2784   \str_clear:N \l__stex_variables_name_str
2785   \str_clear:N \l__stex_variables_args_str
2786   \str_clear:N \l__stex_variables_prec_str
2787   \str_clear:N \l__stex_variables_assoctype_str
2788   \str_clear:N \l__stex_variables_bind_str
2789   \tl_clear:N \l__stex_variables_type_tl
2790   \tl_clear:N \l__stex_variables_def_tl
2791   \tl_clear:N \l__stex_variables_op_tl
2792
2793   \keys_set:nn { stex / vardef } { #1 }
2794 }
2795
2796 \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{}} {
2797   \__stex_variables_args:n {#2}
2798   \str_if_empty:NT \l__stex_variables_name_str {
2799     \str_set:Nx \l__stex_variables_name_str { #1 }
2800   }
2801   \prop_clear:N \l_tmpa_prop
2802   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2803

```

```

2804 \int_zero:N \l_tmpb_int
2805 \bool_set_true:N \l_tmpa_bool
2806 \str_map_inline:Nn \l__stex_variables_args_str {
2807   \token_case_meaning:NnF ##1 {
2808     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2809     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2810     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2811     {\tl_to_str:n a} {
2812       \bool_set_false:N \l_tmpa_bool
2813       \int_incr:N \l_tmpb_int
2814     }
2815     {\tl_to_str:n B} {
2816       \bool_set_false:N \l_tmpa_bool
2817       \int_incr:N \l_tmpb_int
2818     }
2819   }{
2820     \msg_error:nnxx{stex}{error/wrongargs}{
2821       variable~\l__stex_variables_name_str
2822     }{##1}
2823   }
2824 }
2825 \bool_if:NTF \l_tmpa_bool {
2826   % possibly numeric
2827   \str_if_empty:NTF \l__stex_variables_args_str {
2828     \prop_put:Nnn \l_tmpa_prop { args } {}
2829     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2830   }{
2831     \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2832     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2833     \str_clear:N \l_tmpa_str
2834     \int_step_inline:nn \l_tmpa_int {
2835       \str_put_right:Nn \l_tmpa_str i
2836     }
2837     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2838     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2839   }
2840 } {
2841   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2842   \prop_put:Nnx \l_tmpa_prop { arity }
2843     { \str_count:N \l__stex_variables_args_str }
2844 }
2845 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2846 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2847
2848 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop} \l_tmpa_prop
2849
2850 \tl_if_empty:NF \l__stex_variables_op_tl {
2851   \cs_set:cpx {
2852     stex_var_op_notation_ \l__stex_variables_name_str _cs
2853   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2854 }
2855
2856 \tl_set:Nn \l_stex_notation_after_do_tl {
2857   \exp_args:Nne \use:nn {

```

```

2858 \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2859 \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2860 } {{
2861 \exp_after:wN \exp_after:wN \exp_after:wN
2862 \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2863 { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2864 }}
2865 \stex_if_do_html:T {
2866 \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2867 \stex_annotate_invisible:nnn { precedence }
2868 { \l__stex_variables_prec_str }}{
2869 \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2870 \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2871 \stex_annotate_invisible:nnn{macroname}{#1}{
2872 \tl_if_empty:NF \l__stex_variables_def_tl {
2873 \stex_annotate_invisible:nnn{definiens}{
2874 {\l__stex_variables_def_tl$}
2875 }
2876 \str_if_empty:NF \l__stex_variables_assoctype_str {
2877 \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2878 }
2879 \str_if_empty:NF \l__stex_variables_bind_str {
2880 \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{
2881 }
2882 \int_zero:N \l_tmpa_int
2883 \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2884 \tl_clear:N \l_tmpa_tl
2885 \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{{
2886 \int_incr:N \l_tmpa_int
2887 \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2888 \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables_remaining_args_str }
2889 \str_if_eq:VnTF \l_tmpb_str a {
2890 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2891 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2892 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2893 } }
2894 }}{
2895 \str_if_eq:VnTF \l_tmpb_str B {
2896 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2897 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{
2898 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{
2899 } }
2900 }}{
2901 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2902 \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{
2903 } }
2904 }
2905 }
2906 }
2907 \stex_annotate_invisible:nnn { notationcomp }{{
2908 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2909 $ \exp_args:Nno \use:nn { \use:c {
2910 stex_var_notation_\l__stex_variables_name_str _cs
2911 } } { \l_tmpa_tl } $

```



```

2912     }
2913   }
2914   }\ignorespacesandpars
2915 }
2916
2917 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2918 }
2919
2920 \cs_new:Nn \_stex_reset:N {
2921   \tl_if_exist:NTF #1 {
2922     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2923   }{
2924     \let \exp_not:N #1 \exp_not:N \undefined
2925   }
2926 }
2927
2928 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2929   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2930   \exp_args:Nnx \use:nn {
2931     % TODO
2932     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2933       #2
2934     }
2935   }{
2936     \_stex_reset:N \varnot
2937     \_stex_reset:N \vartype
2938     \_stex_reset:N \vardefi
2939   }
2940 }
2941
2942 \NewDocumentCommand \vardef { s } {
2943   \IfBooleanTF#1 {
2944     \__stex_variables_do_complex:nn
2945   }{
2946     \__stex_variables_do_simple:nnn
2947   }
2948 }
2949
2950 \NewDocumentCommand \svar { 0{} m }{
2951   \tl_if_empty:nTF {#1}{
2952     \str_set:Nn \l_tmpa_str { #2 }
2953   }{
2954     \str_set:Nn \l_tmpa_str { #1 }
2955   }
2956   \_stex_term_omv:nn {
2957     var://\l_tmpa_str
2958   }{
2959     \exp_args:Nnx \use:nn {
2960       \def\comp{\_varcomp}
2961       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2962       \comp{ #2 }
2963     }{
2964       \_stex_reset:N \comp
2965       \_stex_reset:N \l_stex_current_symbol_str

```

```

2966     }
2967   }
2968 }
2969
2970
2971
2972 \keys_define:nn { stex / varseq } {
2973   name      .str_set_x:N = \l__stex_variables_name_str ,
2974   args      .int_set:N   = \l__stex_variables_args_int ,
2975   type      .tl_set:N    = \l__stex_variables_type_tl  ,
2976   mid       .tl_set:N    = \l__stex_variables_mid_tl   ,
2977   bind      .choices:nn =
2978     {forall,exists}
2979     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2980 }
2981
2982 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2983   \str_clear:N \l__stex_variables_name_str
2984   \int_set:Nn \l__stex_variables_args_int 1
2985   \tl_clear:N \l__stex_variables_type_tl
2986   \str_clear:N \l__stex_variables_bind_str
2987
2988   \keys_set:nn { stex / varseq } { #1 }
2989 }
2990
2991 \NewDocumentCommand \varseq {m O{}} m m m){
2992   \__stex_variables_seq_args:n { #2 }
2993   \str_if_empty:NT \l__stex_variables_name_str {
2994     \str_set:Nx \l__stex_variables_name_str { #1 }
2995   }
2996   \prop_clear:N \l_tmpa_prop
2997   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2998
2999   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
3000   \int_compare:nNf {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
3001     \msg_error:nnxx{stex}{error/seqlength}
3002     {\int_use:N \l__stex_variables_args_int}
3003     {\seq_count:N \l_tmpa_seq}
3004   }
3005   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
3006   \int_compare:nNf {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
3007     \msg_error:nnxx{stex}{error/seqlength}
3008     {\int_use:N \l__stex_variables_args_int}
3009     {\seq_count:N \l_tmpb_seq}
3010   }
3011   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
3012   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
3013
3014   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str _cs}
3015     \cs_set:Npn {\int_use:N \l__stex_variables_args_int} { #5 }
3016
3017   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str _cs}}
3018   \int_step_inline:nn \l__stex_variables_args_int {
3019     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }

```

```

3020 }
3021 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
3022 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3023 \tl_if_empty:NF \l__stex_variables_mid_tl {
3024   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
3025   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
3026 }
3027 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3028 \int_step_inline:nn \l__stex_variables_args_int {
3029   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
3030 }
3031 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
3032 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
3033
3034
3035 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
3036
3037 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
3038
3039 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq\l__stex_variables_name_str_cs}}
3040
3041 \int_step_inline:nn \l__stex_variables_args_int {
3042   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
3043     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3044   }}
3045 }
3046
3047 \tl_set:Nx \l_tmpa_tl {
3048   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3049     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3050   }
3051 }
3052
3053 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3054
3055 \exp_args:Nno \use:nn {
3056   \cs_generate_from_arg_count:cNnn {stex_varseq\l__stex_variables_name_str_cs}
3057   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}{\l_tmpa_tl}
3058
3059   \stex_debug:nn{sequences}{New~Sequence:~
3060     \expandafter\meaning\csname stex_varseq\l__stex_variables_name_str_cs\endcsname\\~\\
3061     \prop_to_keyval:N \l_tmpa_prop
3062   }
3063   \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3064     \tl_if_empty:NF \l__stex_variables_type_tl {
3065       \stex_annotate:nnn {type}{}{\seqtype\l__stex_variables_type_tl$}
3066     }
3067     \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{}
3068     \str_if_empty:NF \l__stex_variables_bind_str {
3069       \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{}
3070     }
3071   }}
3072
3073   \prop_set_eq:cN {stex_varseq\l__stex_variables_name_str_prop}\l_tmpa_prop

```

```
3074 \ignorespacesandpars
3075 }
3076
3077 </package>
```

Chapter 30

STEX -Terms Implementation

```
3078 <*package>
3079
3080 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3081
3082 <@@=stex_terms>
3083
3084   Warnings and error messages
3085   \msg_new:nnn{stex}{error/nonotation}{
3086     Symbol~#1~invoked,~but~has~no~notation#2!
3087   }
3088   \msg_new:nnn{stex}{error/notationarg}{
3089     Error~in~parsing~notation~#1
3090   }
3091   \msg_new:nnn{stex}{error/noop}{
3092     Symbol~#1~has~no~operator~notation~for~notation~#2
3093   }
3094   \msg_new:nnn{stex}{error/notallowed}{
3095     Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3096   }
3097   \msg_new:nnn{stex}{error/doubleargument}{
3098     Argument~#1~of~symbol~#2~already~assigned
3099   }
3100   \msg_new:nnn{stex}{error/overarity}{
3101     Argument~#1~invalid~for~symbol~#2~with~arity~#3
3102   }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3102
3103
3104 \bool_new:N \l_stex_allow_semantic_bool
3105 \bool_set_true:N \l_stex_allow_semantic_bool
3106
```

```

3107 \cs_new_protected:Nn \stex_invoke_symbol:n {
3108   \bool_if:NTF \l_stex_allow_semantic_bool {
3109     \str_if_eq:eeF {
3110       \prop_item:cn {
3111         l_stex_symdecl_#1_prop
3112       }{ deprecate }
3113     }{}{
3114       \msg_warning:nxxx{stex}{warning/deprecated}{
3115         Symbol~#1
3116       }{
3117         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3118       }
3119     }
3120     \if_mode_math:
3121       \exp_after:wN \__stex_terms_invoke_math:n
3122     \else:
3123       \exp_after:wN \__stex_terms_invoke_text:n
3124     \fi: { #1 }
3125   }{
3126     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3127   }
3128 }
3129
3130 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3131   \peek_charcode_remove:NTF ! {
3132     \__stex_terms_invoke_op_custom:nn {#1}
3133   }{
3134     \__stex_terms_invoke_custom:nn {#1}
3135   }
3136 }
3137
3138 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3139   \peek_charcode_remove:NTF ! {
3140     % operator
3141     \peek_charcode_remove:NTF * {
3142       % custom op
3143       \__stex_terms_invoke_op_custom:nn {#1}
3144     }{
3145       % op notation
3146       \peek_charcode:NTF [ {
3147         \__stex_terms_invoke_op_notation:nw {#1}
3148       }{
3149         \__stex_terms_invoke_op_notation:nw {#1}[]
3150       }
3151     }
3152   }{
3153     \peek_charcode_remove:NTF * {
3154       \__stex_terms_invoke_custom:nn {#1}
3155       % custom
3156     }{
3157       % normal
3158       \peek_charcode:NTF [ {
3159         \__stex_terms_invoke_notation:nw {#1}
3160       }{

```

```

3161         \_stex_terms_invoke_notation:nw {#1}[]
3162     }
3163 }
3164 }
3165 }
3166
3167
3168 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3169     \exp_args:Nnx \use:nn {
3170         \def\comp{\_comp}
3171         \str_set:Nn \l_stex_current_symbol_str { #1 }
3172         \bool_set_false:N \l_stex_allow_semantic_bool
3173         \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3174             \comp{ #2 }
3175         }
3176     }{
3177         \_stex_reset:N \comp
3178         \_stex_reset:N \l_stex_current_symbol_str
3179         \bool_set_true:N \l_stex_allow_semantic_bool
3180     }
3181 }
3182
3183 \keys_define:nn { stex / terms } {
3184     lang .tl_set_x:N = \l_stex_notation_lang_str ,
3185     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3186     unknown .code:n = \str_set:Nx
3187         \l_stex_notation_variant_str \l_keys_key_str
3188 }
3189
3190 \cs_new_protected:Nn \_stex_terms_args:n {
3191     \str_clear:N \l_stex_notation_lang_str
3192     \str_clear:N \l_stex_notation_variant_str
3193
3194     \keys_set:nn { stex / terms } { #1 }
3195 }
3196
3197 \cs_new_protected:Nn \stex_find_notation:nn {
3198     \_stex_terms_args:n { #2 }
3199     \seq_if_empty:cTF {
3200         l_stex_symdecl_ #1 _notations
3201     } {
3202         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3203     } {
3204         \bool_lazy_all:nTF {
3205             {\str_if_empty_p:N \l_stex_notation_variant_str}
3206             {\str_if_empty_p:N \l_stex_notation_lang_str}
3207         }{
3208             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3209         }{
3210             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3211                 \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3212             }{
3213                 \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3214             }{

```

```

3215         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3216         ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3217         }
3218     }
3219 }
3220 }
3221 }
3222
3223 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3224     \exp_args:Nnx \use:nn {
3225         \def\comp{\_comp}
3226         \str_set:Nn \l_stex_current_symbol_str { #1 }
3227         \stex_find_notation:nn { #1 }{ #2 }
3228         \bool_set_false:N \l_stex_allow_semantic_bool
3229         \cs_if_exist:cTF {
3230             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3231         }{
3232             \_stex_term_oms:nnn { #1 }{
3233                 #1 \c_hash_str \l_stex_notation_variant_str
3234             }{
3235                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3236             }
3237         }{
3238             \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3239                 \cs_if_exist:cTF {
3240                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3241                 }{
3242                     \tl_set:Nx \stex_symbol_after_invokation_tl {
3243                         \_stex_reset:N \comp
3244                         \_stex_reset:N \stex_symbol_after_invokation_tl
3245                         \_stex_reset:N \l_stex_current_symbol_str
3246                         \bool_set_true:N \l_stex_allow_semantic_bool
3247                     }
3248                     \def\comp{\_comp}
3249                     \str_set:Nn \l_stex_current_symbol_str { #1 }
3250                     \bool_set_false:N \l_stex_allow_semantic_bool
3251                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3252                 }{
3253                     \msg_error:nnxx{stex}{error/nonotation}{#1}{
3254                     ~\l_stex_notation_variant_str
3255                     }
3256                 }
3257             }{
3258                 \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3259             }
3260         }
3261     }{
3262         \_stex_reset:N \comp
3263         \_stex_reset:N \l_stex_current_symbol_str
3264         \bool_set_true:N \l_stex_allow_semantic_bool
3265     }
3266 }
3267
3268 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```



```

3269 \stex_find_notation:nn { #1 }{ #2 }
3270 \cs_if_exist:cTF {
3271   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3272 }{
3273   \tl_set:Nx \stex_symbol_after_invokation_tl {
3274     \_stex_reset:N \comp
3275     \_stex_reset:N \stex_symbol_after_invokation_tl
3276     \_stex_reset:N \l_stex_current_symbol_str
3277     \bool_set_true:N \l_stex_allow_semantic_bool
3278   }
3279   \def\comp{\_comp}
3280   \str_set:Nn \l_stex_current_symbol_str { #1 }
3281   \bool_set_false:N \l_stex_allow_semantic_bool
3282   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3283 }{
3284   \msg_error:nnxx{stex}{error/nonotation}{#1}{
3285     ~\l_stex_notation_variant_str
3286   }
3287 }
3288 }
3289
3290 \prop_new:N \l__stex_terms_custom_args_prop
3291
3292 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3293   \exp_args:Nnx \use:nn {
3294     \bool_set_false:N \l_stex_allow_semantic_bool
3295     \def\comp{\_comp}
3296     \str_set:Nn \l_stex_current_symbol_str { #1 }
3297     \prop_clear:N \l__stex_terms_custom_args_prop
3298     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3299     \prop_get:cnN {
3300       l_stex_symdecl_#1 _prop
3301     }{ args } \l_tmpa_str
3302     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3303     \tl_set:Nn \arg { \__stex_terms_arg: }
3304     \str_if_empty:NTF \l_tmpa_str {
3305       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3306     }{
3307       \str_if_in:NnTF \l_tmpa_str b {
3308         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3309       }{
3310         \str_if_in:NnTF \l_tmpa_str B {
3311           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3312         }{
3313           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3314         }
3315       }
3316     }
3317     % TODO check that all arguments exist
3318   }{
3319     \_stex_reset:N \l_stex_current_symbol_str
3320     \_stex_reset:N \arg
3321     \_stex_reset:N \comp
3322     \_stex_reset:N \l__stex_terms_custom_args_prop

```

```

3323     \bool_set_true:N \l_stex_allow_semantic_bool
3324   }
3325 }
3326
3327 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3328   \tl_if_empty:nTF {#2}{
3329     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3330     \bool_set_true:N \l_tmpa_bool
3331     \bool_do_while:Nn \l_tmpa_bool {
3332       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3333       \int_incr:N \l_tmpa_int
3334     }{
3335       \bool_set_false:N \l_tmpa_bool
3336     }
3337   }
3338   ){
3339     \int_set:Nn \l_tmpa_int { #2 }
3340   }
3341   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3342   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3343     \msg_error:nnxxx{stex}{error/overarity}
3344     {\int_use:N \l_tmpa_int}
3345     {\l_stex_current_symbol_str}
3346     {\str_count:N \l_tmpa_str}
3347   }
3348   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3349   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3350     \bool_lazy_any:nF {
3351       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3352       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3353     }{
3354       \msg_error:nnxx{stex}{error/doubleargument}
3355       {\int_use:N \l_tmpa_int}
3356       {\l_stex_current_symbol_str}
3357     }
3358   }
3359   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3360   \bool_set_true:N \l_stex_allow_semantic_bool
3361   \IfBooleanTF#1{
3362     \stex_annotate_invisible:n { %TODO
3363       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3364     }
3365   }{ %TODO
3366     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3367   }
3368   \bool_set_false:N \l_stex_allow_semantic_bool
3369 }
3370
3371
3372 \cs_new_protected:Nn \_stex_term_arg:nn {
3373   \bool_set_true:N \l_stex_allow_semantic_bool
3374   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3375   \bool_set_false:N \l_stex_allow_semantic_bool
3376 }

```

```

3377
3378 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3379   \exp_args:Nnx \use:nn
3380   { \int_set:Nn \l__stex_terms_downprec { #2 }
3381     \_stex_term_arg:nn { #1 }{ #3 }
3382   }
3383   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3384 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3385 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3386   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3387   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3388   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3389     \expandafter\if\expandafter\relax\noexpand#3
3390     \expandafter\_stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3391     \else\expandafter\_stex_terms_math_assoc_arg_simple:nn
3392     \expandafter{\expandafter}\expandafter#3\fi
3393   }{
3394     \_stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3395   }
3396 }
3397
3398 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:N {
3399   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3400   \str_if_empty:NTF \l_tmpa_str {
3401     \exp_args:Nx \cs_if_eq:NNTF {
3402       \tl_head:N #1
3403     } \stex_invoke_sequence:n {
3404       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3405       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3406       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\_l_tmpa_str\_prop}{notation}}
3407       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3408       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3409         \exp_not:n{\exp_args:Nnx \use:nn} {
3410           \exp_not:n {
3411             \def\comp{\_varcomp}
3412             \str_set:Nn \l_stex_current_symbol_str
3413             } {varseq://\l_tmpa_str}
3414             \exp_not:n{ ##1 }
3415           }{
3416             \exp_not:n {
3417               \_stex_reset:N \comp
3418               \_stex_reset:N \l_stex_current_symbol_str
3419             }
3420           }
3421         }}}
3422       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3423       \seq_reverse:N \l_tmpa_seq
3424       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3425       \seq_map_inline:Nn \l_tmpa_seq {
3426         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {

```

```

3427         \exp_args:Nno
3428         \l_tmpa_cs { ##1 } \l_tmpa_tl
3429     }
3430 }
3431 \tl_set:Nx \l_tmpa_tl {
3432     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3433         \exp_args:No \exp_not:n \l_tmpa_tl
3434     }
3435 }
3436 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3437 }{
3438     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3439 }
3440 } {
3441     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3442 }
3443
3444 }
3445
3446 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3447     \clist_set:Nn \l_tmpa_clist{ #2 }
3448     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3449         \tl_set:Nn \l_tmpa_tl { #2 }
3450     }{
3451         \clist_reverse:N \l_tmpa_clist
3452         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3453         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3454             \exp_args:No \exp_not:n \l_tmpa_tl
3455         }}
3456         \clist_map_inline:Nn \l_tmpa_clist {
3457             \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3458                 \exp_args:Nno
3459                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3460             }
3461         }
3462     }
3463     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3464 }

```

(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
3465 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3466 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3467 \int_new:N \l__stex_terms_downprec
3468 \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
3469 \tl_set:Nn \l__stex_terms_left_bracket_str (
3470 \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

3471 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3472   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3473     \bool_set_false:N \l__stex_terms_brackets_done_bool
3474     #2
3475   } {
3476     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3477       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3478         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3479         \dobrackets { #2 }
3480       }
3481     }{ #2 }
3482   }
3483 }

(End definition for \__stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

3484 \bool_new:N \l__stex_terms_brackets_done_bool
3485 %\RequirePackage{scalerel}
3486 \cs_new_protected:Npn \dobrackets #1 {
3487   %\ThisStyle{\if D\m@switch
3488   %   \exp_args:Nnx \use:nn
3489   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3490   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3491   % \else
3492   \exp_args:Nnx \use:nn
3493   {
3494     \bool_set_true:N \l__stex_terms_brackets_done_bool
3495     \int_set:Nn \l__stex_terms_downprec \infprec
3496     \l__stex_terms_left_bracket_str
3497     #1
3498   }
3499   {
3500     \bool_set_false:N \l__stex_terms_brackets_done_bool
3501     \l__stex_terms_right_bracket_str
3502     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3503   }
3504   %\fi}
3505 }

```

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

\withbrackets

```

3506 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3507   \exp_args:Nnx \use:nn
3508   {
3509     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }

```

```

3510     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3511     #3
3512   }
3513   {
3514     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3515       {\l__stex_terms_left_bracket_str}
3516     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3517       {\l__stex_terms_right_bracket_str}
3518   }
3519 }

```

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

`\STEXinvisible`

```

3520 \cs_new_protected:Npn \STEXinvisible #1 {
3521   \stex_annotate_invisible:n { #1 }
3522 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3523 \cs_new_protected:Nn \_stex_term_oms:nnn {
3524   \stex_annotate:nnn{ OMID }{ #2 }{
3525     \stex_highlight_term:nn { #1 } { #3 }
3526   }
3527 }
3528
3529 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3530   \_stex_terms_maybe_brackets:nn { #3 }{
3531     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3532   }
3533 }

```

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

`_stex_term_math_omv:nn`

```

3534 \cs_new_protected:Nn \_stex_term_omv:nn {
3535   \stex_annotate:nnn{ OMV }{ #1 }{
3536     \stex_highlight_term:nn { #1 } { #2 }
3537   }
3538 }

```

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

`_stex_term_math_oma:nnnn`

```

3539 \cs_new_protected:Nn \_stex_term_oma:nnn {
3540   \stex_annotate:nnn{ OMA }{ #2 }{
3541     \stex_highlight_term:nn { #1 } { #3 }
3542   }
3543 }
3544
3545 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3546   \_stex_terms_maybe_brackets:nn { #3 }{
3547     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

3548 }
3549 }

```

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

`_stex_term_math_omb:nnnn`

```

3550 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3551   \stex_annotate:nnn{ OMBIND }{ #2 }{
3552     \stex_highlight_term:nn { #1 } { #3 }
3553   }
3554 }
3555
3556 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3557   \__stex_terms_maybe_brackets:nn { #3 }{
3558     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3559   }
3560 }

```

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

`\symref`
`\symname`

```

3561 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3562
3563 \keys_define:nn { stex / symname } {
3564   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3565   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3566   root     .tl_set_x:N = \l__stex_terms_root_tl
3567 }
3568
3569 \cs_new_protected:Nn \stex_symname_args:n {
3570   \tl_clear:N \l__stex_terms_post_tl
3571   \tl_clear:N \l__stex_terms_pre_tl
3572   \tl_clear:N \l__stex_terms_root_str
3573   \keys_set:nn { stex / symname } { #1 }
3574 }
3575
3576 \NewDocumentCommand \symref { m m }{
3577   \let\compemph_uri_prev:\compemph@uri
3578   \let\compemph@uri\symrefemph@uri
3579   \STEXsymbol{#1}!{ #2 }
3580   \let\compemph@uri\compemph_uri_prev:
3581 }
3582
3583 \NewDocumentCommand \synonym { 0{} m m }{
3584   \stex_symname_args:n { #1 }
3585   \let\compemph_uri_prev:\compemph@uri
3586   \let\compemph@uri\symrefemph@uri
3587   % TODO
3588   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3589   \let\compemph@uri\compemph_uri_prev:
3590 }
3591
3592 \NewDocumentCommand \symname { 0{} m }{
3593   \stex_symname_args:n { #1 }
3594   \stex_get_symbol:n { #2 }

```

```

3595 \str_set:Nx \l_tmpa_str {
3596   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3597 }
3598 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3599
3600 \let\compemph_uri_prev:\compemph@uri
3601 \let\compemph@uri\symrefemph@uri
3602 \exp_args:NNx \use:nn
3603 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3604   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3605 } }
3606 \let\compemph@uri\compemph_uri_prev:
3607 }
3608
3609 \NewDocumentCommand \Symname { 0{ } m }{
3610   \stex_symname_args:n { #1 }
3611   \stex_get_symbol:n { #2 }
3612   \str_set:Nx \l_tmpa_str {
3613     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3614   }
3615   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3616   \let\compemph_uri_prev:\compemph@uri
3617   \let\compemph@uri\symrefemph@uri
3618   \exp_args:NNx \use:nn
3619   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3620     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3621     \l__stex_terms_post_tl
3622   } }
3623   \let\compemph@uri\compemph_uri_prev:
3624 }

```

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

30.3 Notation Components

```

3625 <@@=stex_notationcomps>
3626 \cs_new_protected:Nn \stex_highlight_term:nn {
3627   #2
3628 }
3629
3630 \cs_new_protected:Nn \stex_unhighlight_term:n {
3631   % \latexml_if:TF {
3632   %   #1
3633   % } {
3634   %   \rustex_if:TF {
3635   %     #1
3636   %   } {
3637     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3638   %   }
3639   % }
3640 }

```

`\stex_highlight_term:nn`

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

```

\comp
\compemph@uri 3641 \cs_new_protected:Npn \_comp #1 {
\compemph 3642 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3643 \rustex_if:TF {
\defemph@uri 3644 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3645 }{
\symrefemph@uri 3646 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
\varemph 3647 }
\varemph@uri 3648 }
3649 }
3650
3651 \cs_new_protected:Npn \_varcomp #1 {
3652 \str_if_empty:NF \l_stex_current_symbol_str {
3653 \rustex_if:TF {
3654 \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3655 }{
3656 \exp_args:Nnx \varemph@uri { #1 } { \l_stex_current_symbol_str }
3657 }
3658 }
3659 }
3660
3661 \def\comp{\_comp}
3662
3663 \cs_new_protected:Npn \compemph@uri #1 #2 {
3664 \compemph{ #1 }
3665 }
3666
3667
3668 \cs_new_protected:Npn \compemph #1 {
3669 #1
3670 }
3671
3672 \cs_new_protected:Npn \defemph@uri #1 #2 {
3673 \defemph{#1}
3674 }
3675
3676 \cs_new_protected:Npn \defemph #1 {
3677 \textbf{#1}
3678 }
3679
3680 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3681 \symrefemph{#1}
3682 }
3683
3684 \cs_new_protected:Npn \symrefemph #1 {
3685 \textbf{#1}
3686 }
3687
3688 \cs_new_protected:Npn \varemph@uri #1 #2 {
3689 \varemph{#1}
3690 }
3691

```

```

3692 \cs_new_protected:Npn \varemp #1 {
3693     #1
3694 }

```

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

`\ellipses`

```

3695 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3696 \bool_new:N \l_stex_inarray_bool
3697 \bool_set_false:N \l_stex_inarray_bool
3698 \NewDocumentCommand \parray { m m } {
3699     \begingroup
3700     \bool_set_true:N \l_stex_inarray_bool
3701     \begin{array}{#1}
3702         #2
3703     \end{array}
3704 \endgroup
3705 }
3706
3707 \NewDocumentCommand \prmatrix { m } {
3708     \begingroup
3709     \bool_set_true:N \l_stex_inarray_bool
3710     \begin{matrix}
3711         #1
3712     \end{matrix}
3713 \endgroup
3714 }
3715
3716 \def \maybepline {
3717     \bool_if:NT \l_stex_inarray_bool {\hline}
3718 }
3719
3720 \def \parrayline #1 #2 {
3721     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3722 }
3723
3724 \def \pmrow #1 { \parrayline{ }{ #1 } }
3725
3726 \def \parraylineh #1 #2 {
3727     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3728 }
3729
3730 \def \parraycell #1 {
3731     #1 \bool_if:NT \l_stex_inarray_bool {&}
3732 }

```

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

30.4 Variables

3733 <@@=stex_variables>

\stex_invoke_variable:n Invokes a variable

```

3734 \cs_new_protected:Nn \stex_invoke_variable:n {
3735   \if_mode_math:
3736     \exp_after:wN \__stex_variables_invoke_math:n
3737   \else:
3738     \exp_after:wN \__stex_variables_invoke_text:n
3739   \fi: {#1}
3740 }
3741
3742 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3743   %TODO
3744 }
3745
3746 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3747   \peek_charcode_remove:NTF ! {
3748     \peek_charcode_remove:NTF ! {
3749       \peek_charcode:NTF [ {
3750         \__stex_variables_invoke_op_custom:nw
3751       }{
3752         % TODO throw error
3753       }
3754     }{
3755       \__stex_variables_invoke_op:n { #1 }
3756     }
3757   }{
3758     \peek_charcode_remove:NTF * {
3759       \__stex_variables_invoke_text:n { #1 }
3760     }{
3761       \__stex_variables_invoke_math_ii:n { #1 }
3762     }
3763   }
3764 }
3765 }
3766
3767 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3768   \cs_if_exist:cTF {
3769     stex_var_op_notation_ #1 _cs
3770   }{
3771     \exp_args:Nnx \use:nn {
3772       \def\comp{\_varcomp}
3773       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3774       \stex_term_omv:nn { var://#1 }{
3775         \use:c{stex_var_op_notation_ #1 _cs }
3776       }
3777     }{
3778       \stex_reset:N \comp
3779       \stex_reset:N \l_stex_current_symbol_str
3780     }
3781   }{
3782     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{

```

```

3783     \stex_variables_invoke_math_ii:n {#1}
3784   }{
3785     \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
3786   }
3787 }
3788 }
3789
3790 \cs_new_protected:Npn \stex_variables_invoke_math_ii:n #1 {
3791   \cs_if_exist:cTF {
3792     stex_var_notation_#1_cs
3793   }{
3794     \tl_set:Nx \stex_symbol_after_invokation_tl {
3795       \stex_reset:N \comp
3796       \stex_reset:N \stex_symbol_after_invokation_tl
3797       \stex_reset:N \l_stex_current_symbol_str
3798       \bool_set_true:N \l_stex_allow_semantic_bool
3799     }
3800     \def\comp{\_varcomp}
3801     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3802     \bool_set_false:N \l_stex_allow_semantic_bool
3803     \use:c{stex_var_notation_#1_cs}
3804   }{
3805     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3806   }
3807 }

```

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

30.5 Sequences

```

3808 <@@=stex_sequences>
3809
3810 \cs_new_protected:Nn \stex_invoke_sequence:n {
3811   \peek_charcode_remove:NTF ! {
3812     \stex_term_omv:nn {varseq://#1}{
3813       \exp_args:Nnx \use:nn {
3814         \def\comp{\_varcomp}
3815         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3816         \prop_item:cn{stex_varseq_#1_prop}{notation}
3817       }{
3818         \stex_reset:N \comp
3819         \stex_reset:N \l_stex_current_symbol_str
3820       }
3821     }
3822   }{
3823     \bool_set_false:N \l_stex_allow_semantic_bool
3824     \def\comp{\_varcomp}
3825     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3826     \tl_set:Nx \stex_symbol_after_invokation_tl {
3827       \stex_reset:N \comp
3828       \stex_reset:N \stex_symbol_after_invokation_tl
3829       \stex_reset:N \l_stex_current_symbol_str
3830       \bool_set_true:N \l_stex_allow_semantic_bool
3831     }

```

```
3832     \use:c { stex_varseq_#1_cs }
3833   }
3834 }
3835 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
3836 ⟨*package⟩
3837
3838 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3839
3840 Warnings and error messages
3841 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3842   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3843 }
3844 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
3845   Symbol~#1~not~assigned~in~interpretmodule~#2
3846 }
3847 \msg_new:nnn{stex}{error/unknownstructure}{
3848   No~structure~#1~found!
3849 }
3850
3851 \msg_new:nnn{stex}{error/unknownfield}{
3852   No~field~#1~in~instance~#2~found!\#3
3853 }
3854
3855 \msg_new:nnn{stex}{error/keyval}{
3856   Invalid~key=value~pair~#1
3857 }
3858 \msg_new:nnn{stex}{error/instantiate/missing}{
3859   Assignments~missing~in~instantiate:~#1
3860 }
3861 \msg_new:nnn{stex}{error/incompatible}{
3862   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3863 }
3864
```

31.1 Imports with modification

```

3865 <@@=stex_copymodule>
3866 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3867   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3868     \tl_set:Nn \l_tmpa_tl { #1 }
3869     \__stex_copymodule_get_symbol_from_cs:
3870   }{
3871     % argument is a string
3872     % is it a command name?
3873     \cs_if_exist:cTF { #1 }{
3874       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3875       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3876       \str_if_empty:NNTF \l_tmpa_str {
3877         \exp_args:Nx \cs_if_eq:NNTF {
3878           \tl_head:N \l_tmpa_tl
3879         } \stex_invoke_symbol:n {
3880           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3881         }{
3882           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3883         }
3884       } {
3885         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3886       }
3887     }{
3888       % argument is not a command name
3889       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3890       % \l_stex_all_symbols_seq
3891     }
3892   }
3893 }
3894
3895 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3896   \str_set:Nn \l_tmpa_str { #1 }
3897   \bool_set_false:N \l_tmpa_bool
3898   \bool_if:NF \l_tmpa_bool {
3899     \tl_set:Nn \l_tmpa_tl {
3900       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3901     }
3902     \str_set:Nn \l_tmpa_str { #1 }
3903     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3904     \seq_map_inline:Nn #2 {
3905       \str_set:Nn \l_tmpb_str { ##1 }
3906       \str_if_eq:eeT { \l_tmpa_str } {
3907         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3908       } {
3909         \seq_map_break:n {
3910           \tl_set:Nn \l_tmpa_tl {
3911             \str_set:Nn \l_stex_get_symbol_uri_str {
3912               ##1
3913             }
3914           }
3915         }
3916       }

```

```

3917     }
3918     \l_tmpa_tl
3919   }
3920 }
3921
3922 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3923   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3924     { \tl_tail:N \l_tmpa_tl }
3925   \tl_if_single:NTF \l_tmpa_tl {
3926     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3927       \exp_after:wN \str_set:Nn \exp_after:wN
3928         \l_stex_get_symbol_uri_str \l_tmpa_tl
3929       \__stex_copymodule_get_symbol_check:n { #1 }
3930     }{
3931       % TODO
3932       % tail is not a single group
3933     }
3934   }{
3935     % TODO
3936     % tail is not a single group
3937   }
3938 }
3939
3940 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3941   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3942     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3943       :~\seq_use:Nn #1 {,~}
3944     }
3945   }
3946 }
3947
3948 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3949   \stex_import_module_uri:nn { #1 } { #2 }
3950   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3951   \stex_import_require_module:nnnn
3952     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3953     { \l_stex_import_path_str } { \l_stex_import_name_str }
3954   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3955   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3956   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3957   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3958     \seq_map_inline:cn {c_stex_module_###_constants}{
3959       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3960         ##1 ? ####1
3961       }
3962     }
3963   }
3964   \seq_clear:N \l_tmpa_seq
3965   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3966     name      = \l_stex_current_copymodule_name_str ,
3967     module    = \l_stex_current_module_str ,
3968     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3969     includes  = \l_tmpa_seq ,
3970     fields    = \l_tmpa_seq

```



```

3971 }
3972 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3973   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3974   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3975 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3976 \stex_if_smsmode:F {
3977   \begin{stex_annotate_env} {#4} {
3978     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3979   }
3980   \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}}
3981 }
3982 %\bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3983 %\bool_set_false:N \_stex_html_do_output_bool
3984 }
3985 \cs_new_protected:Nn \stex_copymodule_end:n {
3986   \def \l_tmpa_cs ##1 ##2 {#1}
3987   %\bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3988   \tl_clear:N \l_tmpa_tl
3989   \tl_clear:N \l_tmpb_tl
3990   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3991   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3992     \seq_map_inline:cn {c_stex_module_##1_constants}{
3993       \tl_clear:N \l_tmpc_tl
3994       \l_tmpa_cs{##1}{####1}
3995       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3996         \tl_put_right:Nx \l_tmpa_tl {
3997           \prop_set_from_keyval:cn {
3998             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
3999           }}{
4000             \exp_after:wN \prop_to_keyval:N \csname
4001               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_
4002             \endcsname
4003           }
4004           \seq_clear:c {
4005             l_stex_symdecl_
4006             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name
4007             _notations
4008           }
4009         }
4010         \tl_put_right:Nx \l_tmpc_tl {
4011           \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_co
4012           \stex_if_smsmode:F{\stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_c
4013         }
4014         \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodul
4015         \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4016           \tl_put_right:Nx \l_tmpc_tl {
4017             \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymo
4018           }
4019           \tl_put_right:Nx \l_tmpa_tl {
4020             \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4021               \stex_invoke_symbol:n {
4022                 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_
4023             }
4024           }

```

```

4025     }
4026   }
4027 }{
4028   \tl_put_right:Nx \l_tmpc_tl {
4029     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4030   }
4031   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
4032   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
4033   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
4034   \tl_put_right:Nx \l_tmpa_tl {
4035     \prop_set_from_keyval:cn {
4036       l_stex_symdecl_\l_stex_current_module_str ? \l_stex_current_copymodule_name_str
4037     }{
4038       \prop_to_keyval:N \l_tmpa_prop
4039     }
4040     \seq_clear:c {
4041       l_stex_symdecl_
4042       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
4043       _notations
4044     }
4045   }
4046   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4047   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4048     \tl_put_right:Nx \l_tmpc_tl {
4049       \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4050     }
4051     \tl_put_right:Nx \l_tmpa_tl {
4052       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4053         \stex_invoke_symbol:n {
4054           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
4055         }
4056       }
4057     }
4058   }
4059 }
4060 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4061   \tl_put_right:Nx \l_tmpc_tl {
4062     \stex_if_smsmode:F{
4063       $\stex_annotate_invisible:nnn{definiens}}{\exp_after:wN \exp_not:N\csname l__stex_copymodule_copymodule_##1?####1_def_tl\endcsname}
4064     }
4065   }
4066 }
4067 \tl_put_right:Nx \l_tmpb_tl {
4068   \stex_if_smsmode:TF{
4069     \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl}
4070   }{
4071     \stex_annotate:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \exp_after:wN {\l_tmpc_tl} }
4072   }
4073 }
4074 }
4075 }
4076 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4077 \tl_put_left:Nx \l_tmpa_tl {
4078   \prop_set_from_keyval:cn {

```

```

4079     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _pro
4080   }{
4081     \prop_to_keyval:N \l_stex_current_copymodule_prop
4082   }
4083 }
4084 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4085   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4086 }
4087 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4088 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
4089 \exp_args:Nx \stex_do_up_to_module:n {
4090   \exp_args:No \exp_not:n \l_tmpa_tl
4091 }
4092 \stex_debug:nn{copymodule}{output:\meaning \l_tmpb_tl}
4093 \l_tmpb_tl
4094 \stex_if_smsmode:F {
4095   \end{stex_annotate_env}
4096 }
4097 }
4098
4099 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4100   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4101   \stex_deactivate_macro:Nn \symdecl {module~environments}
4102   \stex_deactivate_macro:Nn \symdef {module~environments}
4103   \stex_deactivate_macro:Nn \notation {module~environments}
4104   \stex_reactivate_macro:N \assign
4105   \stex_reactivate_macro:N \renamedekl
4106   \stex_reactivate_macro:N \donotcopy
4107   \stex_smsmode_do:
4108 }{
4109   \stex_copymodule_end:n {}
4110 }
4111
4112 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4113   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4114   \stex_deactivate_macro:Nn \symdecl {module~environments}
4115   \stex_deactivate_macro:Nn \symdef {module~environments}
4116   \stex_deactivate_macro:Nn \notation {module~environments}
4117   \stex_reactivate_macro:N \assign
4118   \stex_reactivate_macro:N \renamedekl
4119   \stex_reactivate_macro:N \donotcopy
4120   \stex_smsmode_do:
4121 }{
4122   \stex_copymodule_end:n {
4123     \tl_if_exist:cF {
4124       l__stex_copymodule_copymodule_##1?##2_def_tl
4125     }{
4126       \str_if_eq:eeF {
4127         \prop_item:cn{
4128           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4129       }{ true }{
4130         \msg_error:nxxx{stex}{error/interpretmodule/noddefinens}{
4131           ##1?##2
4132         }\l_stex_current_copymodule_name_str}

```

```

4133     }
4134   }
4135 }
4136 }
4137
4138 \NewDocumentCommand \donotcopy { m }{
4139   \str_clear:N \l_stex_import_name_str
4140   \str_set:Nn \l_tmpa_str { #1 }
4141   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4142   \seq_map_inline:Nn \l_stex_all_modules_seq {
4143     \str_set:Nn \l_tmpb_str { ##1 }
4144     \str_if_eq:eeT { \l_tmpa_str } {
4145       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4146     } {
4147       \seq_map_break:n {
4148         \stex_if_do_html:T {
4149           \stex_if_smsmode:F {
4150             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4151               \stex_annotate:nnn{domain}{##1}{}}
4152           }
4153         }
4154       }
4155       \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4156     }
4157   }
4158   \seq_map_inline:cn {c_stex_module_##1_copymodules}{
4159     \str_set:Nn \l_tmpb_str { #####1 }
4160     \str_if_eq:eeT { \l_tmpa_str } {
4161       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4162     } {
4163       \seq_map_break:n {\seq_map_break:n {
4164         \stex_if_do_html:T {
4165           \stex_if_smsmode:F {
4166             \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4167               \stex_annotate:nnn{domain}{
4168                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4169               }{}
4170             }
4171           }
4172         }
4173         \str_set:Nx \l_stex_import_name_str {
4174           \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4175         }
4176       }}
4177     }
4178   }
4179 }
4180 \str_if_empty:NTF \l_stex_import_name_str {
4181   % TODO throw error
4182 }{
4183   \stex_collect_imports:n {\l_stex_import_name_str }
4184   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4185     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4186     \seq_map_inline:cn {c_stex_module_##1_constants}{

```

```

4187 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4188 \bool_lazy_any:nT {
4189   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4190   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}
4191   { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_def_tl}}
4192 }{
4193   % TODO throw error
4194 }
4195 }
4196 }
4197 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4198 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4199 \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4200 }
4201 \stex_smsmode_do:
4202 }
4203
4204 \NewDocumentCommand \assign { m m }{
4205   \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4206   \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4207   \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4208   \stex_smsmode_do:
4209 }
4210
4211 \keys_define:nn { stex / renamedec1 } {
4212   name .str_set_x:N = \l_stex_renamedec1_name_str
4213 }
4214 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4215   \str_clear:N \l_stex_renamedec1_name_str
4216   \keys_set:nn { stex / renamedec1 } { #1 }
4217 }
4218
4219 \NewDocumentCommand \renamedec1 { 0{} m m }{
4220   \__stex_copymodule_renamedec1_args:n { #1 }
4221   \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4222   \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4223   \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4224   \str_if_empty:NTF \l_stex_renamedec1_name_str {
4225     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4226       \l_stex_get_symbol_uri_str
4227     } }
4228   } {
4229     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4230     \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4231     \prop_set_eq:cc {l_stex_symdecl_
4232       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4233       _prop
4234     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4235     \seq_set_eq:cc {l_stex_symdecl_
4236       \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4237       _notations
4238     }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4239     \prop_put:cnx {l_stex_symdecl_
4240       \l_stex_current_module_str ? \l_stex_renamedec1_name_str

```

```

4241   _prop
4242   ){ name }{ \l_stex_renamedekl_name_str }
4243   \prop_put:cnx {l_stex_symdecl_
4244     \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4245     _prop
4246     }{ module }{ \l_stex_current_module_str }
4247     \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4248       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4249     }
4250     \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4251       \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4252     } }
4253   }
4254   \stex_smsmode_do:
4255 }
4256
4257 \stex_deactivate_macro:Nn \assign {copymodules}
4258 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4259 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4260
4261
4262 \seq_new:N \l_stex_implicit_morphisms_seq
4263 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4264   \stex_import_module_uri:nn { #1 } { #2 }
4265   \stex_debug:nn{implicits}{
4266     Implicit~morphism:~
4267     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4268   }
4269   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4270     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4271   }{
4272     \msg_error:nnn{stex}{error/conflictingmodules}{
4273       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4274     }
4275   }
4276
4277   % TODO
4278
4279
4280
4281   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4282     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4283   }
4284 }
4285

```

31.2 The feature environment

structural@feature

```

4286 <@@=stex_features>
4287
4288 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4289   \stex_if_in_module:F {

```

```

4290 \msg_set:nn{stex}{error/nomodule}{
4291   Structural~Feature~has~to~occur~in~a~module:\\
4292   Feature~#2~of~type~#1\\
4293   In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4294 }
4295 \msg_error:nn{stex}{error/nomodule}
4296 }
4297
4298 \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str
4299
4300 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4301
4302 \stex_if_smsmode:F {
4303   \begin{stex_annotate_env}{ feature:#1 }{\l_tmpa_str ? #2 - #1}
4304   \stex_annotate_invisible:nn{header}{}{ #3 }
4305 }
4306 }{
4307   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4308   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4309   \stex_debug:nn{features}{
4310     Feature: \l_stex_last_feature_str
4311   }
4312   \stex_if_smsmode:F {
4313     \end{stex_annotate_env}
4314   }
4315 }

```

31.3 Structure

structure

```

4316 <@@=stex_structures>
4317 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4318   \prop_if_exist:cF {c_stex_module_\l_stex_current_module_str _structures}{
4319     \prop_new:c {c_stex_module_\l_stex_current_module_str _structures}
4320   }
4321   \prop_gput:cxn{c_stex_module_\l_stex_current_module_str _structures}
4322   {#1}{#2}
4323 }
4324
4325 \keys_define:nn { stex / features / structure } {
4326   name .str_set_x:N = \l__stex_structures_name_str ,
4327 }
4328
4329 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4330   \str_clear:N \l__stex_structures_name_str
4331   \keys_set:nn { stex / features / structure } { #1 }
4332 }
4333
4334 \NewDocumentEnvironment{mathstructure}{m O{}}{
4335   \__stex_structures_structure_args:n { #2 }
4336   \str_if_empty:NT \l__stex_structures_name_str {
4337     \str_set:Nx \l__stex_structures_name_str { #1 }
4338   }

```

```

4339 \stex_suppress_html:n {
4340   \exp_args:Nx \stex_symdecl_do:nn {
4341     name = \l__stex_structures_name_str ,
4342     def = {\STEXsymbol{module-type}}{
4343       \stex_term_math_oms:nnnn {
4344         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4345         { ns } ?
4346         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4347         { name } / \l__stex_structures_name_str - structure
4348       }{}{}{}
4349     }}
4350   }{ #1 }
4351 }
4352 \exp_args:Nnnx
4353 \begin{structural_feature_module}{ structure }
4354 { \l__stex_structures_name_str }{}
4355 \stex_smsmode_do:
4356 {}{
4357   \end{structural_feature_module}
4358   \stex_reset_up_to_module:n \l_stex_last_feature_str
4359   \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4360   \seq_clear:N \l_tmpa_seq
4361   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4362     \seq_map_inline:cn{c_stex_module_##1_constants}{
4363       \seq_put_right:Nn \l_tmpa_seq { ##1 ? ###1 }
4364     }
4365   }
4366   \exp_args:Nnno
4367   \prop_gput:cnn {c_stex_module_\l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4368   \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4369   \stex_add_structure_to_current_module:nn
4370     \l__stex_structures_name_str
4371     \l_stex_last_feature_str
4372   \exp_args:Nx
4373   \stex_add_to_current_module:n {
4374     \tl_set:cn { #1 }{
4375       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4376     }
4377   }
4378   \exp_args:Nx
4379   \stex_do_up_to_module:n {
4380     \tl_set:cn { #1 }{
4381       \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structure
4382     }
4383   }
4384 }
4385
4386 \cs_new:Nn \stex_invoke_structure:nn {
4387   \stex_invoke_symbol:n { #1?#2 }
4388 }
4389
4390 \cs_new_protected:Nn \stex_get_structure:n {
4391   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4392     \tl_set:Nn \l_tmpa_tl { #1 }

```



```

4393     \__stex_structures_get_from_cs:
4394   }{
4395     \cs_if_exist:cTF { #1 }{
4396       \cs_set_eq:Nc \l_tmpa_cs { #1 }
4397       \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4398       \str_if_empty:NTF \l_tmpa_str {
4399         \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4400           \__stex_structures_get_from_cs:
4401         }{
4402           \__stex_structures_get_from_string:n { #1 }
4403         }
4404       }{
4405         \__stex_structures_get_from_string:n { #1 }
4406       }
4407     }{
4408       \__stex_structures_get_from_string:n { #1 }
4409     }
4410   }
4411 }
4412
4413 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4414   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4415     { \tl_tail:N \l_tmpa_tl }
4416   \str_set:Nx \l_tmpa_str {
4417     \exp_after:wN \use_i:nn \l_tmpa_tl
4418   }
4419   \str_set:Nx \l_tmpb_str {
4420     \exp_after:wN \use_ii:nn \l_tmpa_tl
4421   }
4422   \str_set:Nx \l_stex_get_structure_str {
4423     \l_tmpa_str ? \l_tmpb_str
4424   }
4425   \str_set:Nx \l_stex_get_structure_module_str {
4426     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4427   }
4428 }
4429
4430 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4431   \tl_set:Nn \l_tmpa_tl {
4432     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4433   }
4434   \str_set:Nn \l_tmpa_str { #1 }
4435   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4436
4437   \seq_map_inline:Nn \l_stex_all_modules_seq {
4438     \prop_if_exist:cT {c_stex_module_##1_structures} {
4439       \prop_map_inline:cn {c_stex_module_##1_structures} {
4440         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4441           \prop_map_break:n{ \seq_map_break:n{
4442             \tl_set:Nn \l_tmpa_tl {
4443               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4444               \str_set:Nn \l_stex_get_structure_module_str {####2}
4445             }
4446           }}

```

```

4447     }
4448   }
4449 }
4450 }
4451 \l_tmpa_tl
4452 }

```

\instantiate

```

4453
4454 \keys_define:nn { stex / instantiate } {
4455   name .str_set_x:N = \l__stex_structures_name_str
4456 }
4457 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4458   \str_clear:N \l__stex_structures_name_str
4459   \keys_set:nn { stex / instantiate } { #1 }
4460 }
4461
4462 \NewDocumentCommand \instantiate {m O{} m m m}{
4463   \begingroup
4464     \stex_get_structure:n {#4}
4465     \__stex_structures_instantiate_args:n { #2 }
4466     \str_if_empty:NT \l__stex_structures_name_str {
4467       \str_set:Nn \l__stex_structures_name_str { #1 }
4468     }
4469     \seq_clear:N \l__stex_structures_fields_seq
4470     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4471     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4472       \seq_map_inline:cn {c_stex_module_##1_constants}{
4473         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4474       }
4475     }
4476     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4477     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4478     \prop_clear:N \l_tmpa_prop
4479     \seq_map_inline:Nn \l_tmpa_seq {
4480       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4481       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4482         \msg_error:nnn{stex}{error/keyval}{##1}
4483       }
4484       \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4485       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4486       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4487       \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4488       \exp_args:Nxx \str_if_eq:nnF
4489         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4490         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4491         \msg_error:nnxxx{stex}{error/incompatible}
4492         {\l__stex_structures_dom_str}
4493         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4494         {\l_stex_get_symbol_uri_str}
4495         {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4496       }
4497       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4498     }

```

```

4499 \seq_if_empty:NF \l__stex_structures_fields_seq {
4500   \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
4501 }
4502 \exp_args:Nx
4503 \stex_add_to_current_module:n {
4504   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l_stex_structur
4505     domain = \l_stex_get_structure_module_str ,
4506     \prop_to_keyval:N \l_tmpa_prop
4507   }
4508   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l_stex_structur
4509 }
4510 \exp_args:Nx
4511 \stex_do_up_to_module:n {
4512   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l_stex_structur
4513     domain = \l_stex_get_structure_module_str ,
4514     \prop_to_keyval:N \l_tmpa_prop
4515   }
4516   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4517 }
4518 \stex_debug:nn{instantiate}{
4519   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \\\
4520   \prop_to_keyval:N \l_tmpa_prop
4521 }
4522 \exp_args:Nxx \stex_symdecl_do:nn {
4523   type={\STEXsymbol{module-type}}{
4524     \stex_term_math_oms:nnnn {
4525       \l_stex_get_structure_module_str
4526     }{}{0}{}
4527   }}
4528   {}{\l__stex_structures_name_str}
4529   \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4530 \endgroup
4531 \stex_smsmode_do:\ignorespacesandpars
4532 }
4533
4534 \cs_new_protected:Nn \stex_symbol_or_var:n {
4535   \cs_if_exist:cTF{#1}{
4536     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4537     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4538     \str_if_empty:NTF \l_tmpa_str {
4539       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4540       \stex_invoke_variable:n {
4541         \bool_set_true:N \l_stex_symbol_or_var_bool
4542         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4543         \str_set:Nx \l_stex_get_symbol_uri_str {
4544           \exp_after:wN \use:n \l_tmpa_tl
4545         }
4546       }{
4547         \bool_set_false:N \l_stex_symbol_or_var_bool
4548         \stex_get_symbol:n{#1}
4549       }
4550     }{
4551       \__stex_structures_symbolorvar_from_string:n{ #1 }
4552     }

```

```

4553 }{
4554   \__stex_structures_symbolorvar_from_string:n{ #1 }
4555 }
4556 }
4557
4558 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4559   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4560     \bool_set_true:N \l_stex_symbol_or_var_bool
4561     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4562   }{
4563     \bool_set_false:N \l_stex_symbol_or_var_bool
4564     \stex_get_symbol:n{#1}
4565   }
4566 }
4567
4568 \keys_define:nn { stex / varinstantiate } {
4569   name          .str_set_x:N = \l__stex_structures_name_str,
4570   bind          .choices:nn =
4571     {forall,exists}
4572     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4573 }
4574
4575 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4576   \str_clear:N \l__stex_structures_name_str
4577   \str_clear:N \l__stex_structures_bind_str
4578   \keys_set:nn { stex / varinstantiate } { #1 }
4579 }
4580
4581 \NewDocumentCommand \varinstantiate {m O{} m m m}{
4582   \begin{group}
4583     \stex_get_structure:n {#4}
4584     \__stex_structures_varinstantiate_args:n { #2 }
4585     \str_if_empty:NT \l__stex_structures_name_str {
4586       \str_set:Nn \l__stex_structures_name_str { #1 }
4587     }
4588     \stex_if_do_html:TF{
4589       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4590     }{\use:n}
4591     {
4592       \stex_if_do_html:T{
4593         \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{}
4594       }
4595       \seq_clear:N \l__stex_structures_fields_seq
4596       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4597       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4598         \seq_map_inline:cn {c_stex_module_##1_constants}{
4599           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4600         }
4601       }
4602       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4603       \prop_clear:N \l_tmpa_prop
4604       \tl_if_empty:nF {#3} {
4605         \seq_set_split:Nnn \l_tmpa_seq , {#3}
4606         \seq_map_inline:Nn \l_tmpa_seq {

```

```

4607 \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4608 \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4609     \msg_error:nnn{stex}{error/keyval}{##1}
4610 }
4611 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4612 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4613 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
4614 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4615 \stex_if_do_html:T{
4616     \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4617 }
4618 \bool_if:NTF \l_stex_symbol_or_var_bool {
4619     \exp_args:Nxx \str_if_eq:nnF
4620         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4621         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4622         \msg_error:nnxxx{stex}{error/incompatible}
4623         {\l__stex_structures_dom_str}
4624         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4625         {\l_stex_get_symbol_uri_str}
4626         {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4627     }
4628     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4629 }}{
4630     \exp_args:Nxx \str_if_eq:nnF
4631         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4632         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4633         \msg_error:nnxxx{stex}{error/incompatible}
4634         {\l__stex_structures_dom_str}
4635         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4636         {\l_stex_get_symbol_uri_str}
4637         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4638     }
4639     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4640 }}{
4641 }
4642 }
4643 \tl_gclear:N \g__stex_structures_aftergroup_tl
4644 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4645     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq ##1}
4646     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4647     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4648         \stex_find_notation:nn{##1}{
4649             \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4650                 {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4651             \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4652             \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4653                 \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4654                 {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4655                 \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4656             }
4657         }
4658     }
4659     \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4660         \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{

```

```

4661         name    = \l_tmpa_str ,
4662         args     = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4663         arity    = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4664         assocs   = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4665     }
4666     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4667     {g__stex_structures_tmpa_\l_tmpa_str_cs}
4668     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4669     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4670 }
4671 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4672 }
4673 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4674     \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4675         domain = \l_stex_get_structure_module_str ,
4676         \prop_to_keyval:N \l_tmpa_prop
4677     }
4678     \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4679     \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4680         \exp_args:Nnx \exp_not:N \use:nn {
4681             \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4682             \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4683                 \exp_not:n{
4684                     \_varcomp{#5}
4685                 }
4686             }
4687         }{
4688             \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4689         }
4690     }
4691 }
4692 }
4693 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4694 \aftergroup\g__stex_structures_aftergroup_tl
4695 \endgroup
4696 \stex_smsmode_do:\ignorespacesandpars
4697 }
4698
4699 \cs_new_protected:Nn \stex_invoke_instance:n {
4700     \peek_charcode_remove:NTF ! {
4701         \stex_invoke_symbol:n{#1}
4702     }{
4703         \_stex_invoke_instance:nn {#1}
4704     }
4705 }
4706
4707
4708 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4709     \peek_charcode_remove:NTF ! {
4710         \exp_args:Nnx \use:nn {
4711             \def\comp{\_varcomp}
4712             \use:c{l_stex_varinstance_#1_op_tl}
4713         }{
4714             \_stex_reset:N \comp

```

```

4715     }
4716   }{
4717     \_stex_invoke_varinstance:nn {#1}
4718   }
4719 }
4720
4721 \cs_new_protected:Nn \_stex_invoke_instance:nn {
4722   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4723     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4724   }{
4725     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4726     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4727       \prop_to_keyval:N \l_tmpa_prop
4728     }
4729   }
4730 }
4731
4732 \cs_new_protected:Nn \_stex_invoke_varinstance:nn {
4733   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4734     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4735     \l_tmpa_tl
4736   }{
4737     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4738   }
4739 }

```

(End definition for `\instantiate`. This function is documented on page [31](#).)

`\stex_invoke_structure:nnn`

```

4740 % #1: URI of the instance
4741 % #2: URI of the instantiated module
4742 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4743   \tl_if_empty:nTF{ #3 }{
4744     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4745       c_stex_feature_ #2 _prop
4746     }
4747     \tl_clear:N \l_tmpa_tl
4748     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4749     \seq_map_inline:Nn \l_tmpa_seq {
4750       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4751       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4752       \cs_if_exist:cT {
4753         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4754       }{
4755         \tl_if_empty:NF \l_tmpa_tl {
4756           \tl_put_right:Nn \l_tmpa_tl {,}
4757         }
4758         \tl_put_right:Nx \l_tmpa_tl {
4759           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4760         }
4761       }
4762     }
4763     \exp_args:No \mathstrut \l_tmpa_tl
4764   }{

```

```

4765     \stex_invoke_symbol:n{#1/#3}
4766   }
4767 }

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

4768 \endpackage

```


Chapter 32

STEX -Statements Implementation

```
4769 <*package>
4770
4771 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4772
4773 <@@=stex_statements>
    Warnings and error messages
4774
\titleemph
4775 \def\titleemph#1{\textbf{#1}}
    (End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4776 \keys_define:nn {stex / definiendum }{
4777   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4778   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4779   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4780   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4781 }
4782 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4783   \str_clear:N \l__stex_statements_definiendum_root_str
4784   \tl_clear:N \l__stex_statements_definiendum_post_tl
4785   \str_clear:N \l__stex_statements_definiendum_gfa_str
4786   \keys_set:nn { stex / definiendum }{ #1 }
4787 }
4788 \NewDocumentCommand \definiendum { O{} m m } {
4789   \__stex_statements_definiendum_args:n { #1 }
4790   \stex_get_symbol:n { #2 }
4791   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4792   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4793     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4794     \tl_set:Nn \l_tmpa_tl { #3 }
4795   } {
4796     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4797     \tl_set:Nn \l_tmpa_tl {
4798       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4799     }
4800   }
4801 } {
4802   \tl_set:Nn \l_tmpa_tl { #3 }
4803 }
4804
4805 % TODO root
4806 \rustex_if:TF {
4807   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4808 } {
4809   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4810 }
4811 }
4812 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4813
4814 \NewDocumentCommand \definame { 0{ } m } {
4815   \__stex_statements_definiendum_args:n { #1 }
4816   % TODO: root
4817   \stex_get_symbol:n { #2 }
4818   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4819   \str_set:Nx \l_tmpa_str {
4820     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4821   }
4822   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4823   \rustex_if:TF {
4824     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4825       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4826     }
4827   } {
4828     \exp_args:Nnx \defemph@uri {
4829       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4830     } { \l_stex_get_symbol_uri_str }
4831   }
4832 }
4833 \stex_deactivate_macro:Nn \definame {definition~environments}
4834
4835 \NewDocumentCommand \Definame { 0{ } m } {
4836   \__stex_statements_definiendum_args:n { #1 }
4837   \stex_get_symbol:n { #2 }
4838   \str_set:Nx \l_tmpa_str {
4839     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4840   }
4841   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4842   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4843   \rustex_if:TF {

```

```

4844 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4845 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4846 }
4847 } {
4848 \exp_args:Nnx \defemph@uri {
4849 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4850 } { \l_stex_get_symbol_uri_str }
4851 }
4852 }
4853 \stex_deactivate_macro:Nn \Definame {definition~environments}
4854
4855 \NewDocumentCommand \premise { m }{
4856 \stex_annotate:nnn{ premise }{}{ #1 }
4857 }
4858 \NewDocumentCommand \conclusion { m }{
4859 \stex_annotate:nnn{ conclusion }{}{ #1 }
4860 }
4861 \NewDocumentCommand \definiens { O{} m }{
4862 \str_clear:N \l_stex_get_symbol_uri_str
4863 \tl_if_empty:nF {#1} {
4864 \stex_get_symbol:n { #1 }
4865 }
4866 \str_if_empty:NT \l_stex_get_symbol_uri_str {
4867 \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4868 \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4869 }{
4870 % TODO throw error
4871 }
4872 }
4873 \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
4874 {\l_stex_current_module_str}{
4875 \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4876 }{true}}{
4877 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4878 \exp_args:Nx \stex_add_to_current_module:n {
4879 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4880 }
4881 }
4882 }
4883 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4884 }
4885
4886 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4887 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4888 \stex_deactivate_macro:Nn \definiens {definition~environments}
4889

```

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

sdefinition

```

4890
4891 \keys_define:nn {stex / sdefinition }{
4892 type .str_set_x:N = \sdefinitiontype,
4893 id .str_set_x:N = \sdefinitionid,

```

```

4894 name .str_set_x:N = \sdefinitionname,
4895 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4896 title .tl_set:N = \sdefinitiontitle
4897 }
4898 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4899 \str_clear:N \sdefinitiontype
4900 \str_clear:N \sdefinitionid
4901 \str_clear:N \sdefinitionname
4902 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4903 \tl_clear:N \sdefinitiontitle
4904 \keys_set:nn { stex / sdefinition }{ #1 }
4905 }
4906
4907 \NewDocumentEnvironment{sdefinition}{0{}}{
4908 \__stex_statements_sdefinition_args:n{ #1 }
4909 \stex_reactivate_macro:N \definiendum
4910 \stex_reactivate_macro:N \definame
4911 \stex_reactivate_macro:N \Definame
4912 \stex_reactivate_macro:N \premise
4913 \stex_reactivate_macro:N \definiens
4914 \stex_if_smsmode:F{
4915 \seq_clear:N \l_tmpa_seq
4916 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4917 \tl_if_empty:nF{ ##1 }{
4918 \stex_get_symbol:n { ##1 }
4919 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4920 \l_stex_get_symbol_uri_str
4921 }
4922 }
4923 }
4924 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4925 \exp_args:Nnnx
4926 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4927 \str_if_empty:NF \sdefinitiontype {
4928 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4929 }
4930 \str_if_empty:NF \sdefinitionname {
4931 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4932 }
4933 \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4934 \tl_clear:N \l_tmpa_tl
4935 \clist_map_inline:Nn \l_tmpa_clist {
4936 \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4937 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4938 }
4939 }
4940 \tl_if_empty:NTF \l_tmpa_tl {
4941 \__stex_statements_sdefinition_start:
4942 }{
4943 \l_tmpa_tl
4944 }
4945 }
4946 \stex_ref_new_doc_target:n \sdefinitionid
4947 \stex_smsmode_do:

```

```

4948 }{
4949   \stex_suppress_html:n {
4950     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4951   }
4952   \stex_if_smsmode:F {
4953     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4954     \tl_clear:N \l_tmpa_tl
4955     \clist_map_inline:Nn \l_tmpa_clist {
4956       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4957         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4958       }
4959     }
4960     \tl_if_empty:NTF \l_tmpa_tl {
4961       \__stex_statements_sdefinition_end:
4962     }{
4963       \l_tmpa_tl
4964     }
4965     \end{stex_annotate_env}
4966   }
4967 }

```

\stexpatchdefinition

```

4968 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4969   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4970     ~(\sdefinitiontitle)
4971   }~}
4972 }
4973 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
4974
4975 \newcommand\stexpatchdefinition[3] [] {
4976   \str_set:Nx \l_tmpa_str{ #1 }
4977   \str_if_empty:NTF \l_tmpa_str {
4978     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4979     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4980   }{
4981     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4982     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4983   }
4984 }

```

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

\inlinedef inline:

```

4985 \keys_define:nn {stex / inlinedef }{
4986   type      .str_set_x:N = \sdefinitiontype,
4987   id        .str_set_x:N = \sdefinitionid,
4988   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4989   name      .str_set_x:N = \sdefinitionname
4990 }
4991 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4992   \str_clear:N \sdefinitiontype
4993   \str_clear:N \sdefinitionid
4994   \str_clear:N \sdefinitionname
4995   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

4996 \keys_set:nn { stex / inlinedef }{ #1 }
4997 }
4998 \NewDocumentCommand \inlinedef { 0{} m } {
4999   \begingroup
5000   \__stex_statements_inlinedef_args:n{ #1 }
5001   \stex_reactivate_macro:N \definiendum
5002   \stex_reactivate_macro:N \definame
5003   \stex_reactivate_macro:N \Definame
5004   \stex_reactivate_macro:N \premise
5005   \stex_reactivate_macro:N \definiens
5006   \stex_ref_new_doc_target:n \sdefinitionid
5007   \stex_if_smsmode:TF{\stex_suppress_html:n {
5008     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
5009   }}{
5010     \seq_clear:N \l_tmpa_seq
5011     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
5012       \tl_if_empty:NF{ ##1 }{
5013         \stex_get_symbol:n { ##1 }
5014         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5015           \l_stex_get_symbol_uri_str
5016         }
5017       }
5018     }
5019     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
5020     \exp_args:Nnx
5021     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
5022       \str_if_empty:NF \sdefinitiontype {
5023         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
5024       }
5025       #2
5026       \str_if_empty:NF \sdefinitionname {
5027         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
5028         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
5029       }
5030     }
5031   }
5032   \endgroup
5033   \stex_smsmode_do:
5034 }

```

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

32.2 Assertions

sassertion

```

5035
5036 \keys_define:nn {stex / sassertion }{
5037   type      .str_set_x:N = \sassertiontype,
5038   id        .str_set_x:N = \sassertionid,
5039   title     .tl_set:N     = \sassertiontitle ,
5040   for       .clist_set:N  = \l__stex_statements_sassertion_for_clist ,
5041   name      .str_set_x:N  = \sassertionname
5042 }

```

```

5043 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
5044   \str_clear:N \sassertiontype
5045   \str_clear:N \sassertionid
5046   \str_clear:N \sassertionname
5047   \clist_clear:N \l__stex_statements_sassertion_for_clist
5048   \tl_clear:N \sassertiontitle
5049   \keys_set:nn { stex / sassertion }{ #1 }
5050 }
5051
5052 %\tl_new:N \g__stex_statements_aftergroup_tl
5053
5054 \NewDocumentEnvironment{sassertion}{0{}}{
5055   \__stex_statements_sassertion_args:n{ #1 }
5056   \stex_reactivate_macro:N \premise
5057   \stex_reactivate_macro:N \conclusion
5058   \stex_if_smsmode:F {
5059     \seq_clear:N \l_tmpa_seq
5060     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5061       \tl_if_empty:nF{ ##1 }{
5062         \stex_get_symbol:n { ##1 }
5063         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5064           \l_stex_get_symbol_uri_str
5065         }
5066       }
5067     }
5068     \exp_args:Nnnx
5069     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5070     \str_if_empty:NF \sassertiontype {
5071       \stex_annotate_invisible:nnn{type}{\sassertiontype}{ }
5072     }
5073     \str_if_empty:NF \sassertionname {
5074       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{ }
5075     }
5076     \clist_set:Nn \l_tmpa_clist \sassertiontype
5077     \tl_clear:N \l_tmpa_tl
5078     \clist_map_inline:Nn \l_tmpa_clist {
5079       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5080         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5081       }
5082     }
5083     \tl_if_empty:NTF \l_tmpa_tl {
5084       \__stex_statements_sassertion_start:
5085     }{
5086       \l_tmpa_tl
5087     }
5088   }
5089   \str_if_empty:NTF \sassertionid {
5090     \str_if_empty:NF \sassertionname {
5091       \stex_ref_new_doc_target:n { }
5092     }
5093   } {
5094     \stex_ref_new_doc_target:n \sassertionid
5095   }
5096   \stex_smsmode_do:

```

```

5097 }{
5098   \str_if_empty:NF \sassertionname {
5099     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5100     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5101   }
5102   \stex_if_smsmode:F {
5103     \clist_set:Nn \l_tmpa_clist \sassertiontype
5104     \tl_clear:N \l_tmpa_tl
5105     \clist_map_inline:Nn \l_tmpa_clist {
5106       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5107         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5108       }
5109     }
5110     \tl_if_empty:NTF \l_tmpa_tl {
5111       \__stex_statements_sassertion_end:
5112     }{
5113       \l_tmpa_tl
5114     }
5115     \end{stex_annotate_env}
5116   }
5117 }

```

\stexpatchassertion

```

5118
5119 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5120   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5121     (\sassertiontitle)
5122   }~}
5123 }
5124 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5125
5126 \newcommand\stexpatchassertion[3] [] {
5127   \str_set:Nx \l_tmpa_str{ #1 }
5128   \str_if_empty:NTF \l_tmpa_str {
5129     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5130     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5131   }{
5132     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5133     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5134   }
5135 }

```

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

\inlineass inline:

```

5136 \keys_define:nn {stex / inlineass }{
5137   type      .str_set_x:N = \sassertiontype,
5138   id        .str_set_x:N = \sassertionid,
5139   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5140   name      .str_set_x:N = \sassertionname
5141 }
5142 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5143   \str_clear:N \sassertiontype
5144   \str_clear:N \sassertionid

```



```

5145 \str_clear:N \sassertionname
5146 \clist_clear:N \l__stex_statements_sassertion_for_clist
5147 \keys_set:nn { stex / inlineass }{ #1 }
5148 }
5149 \NewDocumentCommand \inlineass { 0{} m } {
5150 \beginingroup
5151 \stex_reactivate_macro:N \premise
5152 \stex_reactivate_macro:N \conclusion
5153 \__stex_statements_inlineass_args:n{ #1 }
5154 \str_if_empty:NTF \sassertionid {
5155 \str_if_empty:NF \sassertionname {
5156 \stex_ref_new_doc_target:n {}
5157 }
5158 } {
5159 \stex_ref_new_doc_target:n \sassertionid
5160 }
5161
5162 \stex_if_smsmode:TF{
5163 \str_if_empty:NF \sassertionname {
5164 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5165 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5166 }
5167 }{
5168 \seq_clear:N \l_tmpa_seq
5169 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5170 \tl_if_empty:NF{ ##1 }{
5171 \stex_get_symbol:n { ##1 }
5172 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5173 \l_stex_get_symbol_uri_str
5174 }
5175 }
5176 }
5177 \exp_args:Nnx
5178 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5179 \str_if_empty:NF \sassertiontype {
5180 \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5181 }
5182 #2
5183 \str_if_empty:NF \sassertionname {
5184 \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5185 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5186 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5187 }
5188 }
5189 }
5190 \endgroup
5191 \stex_smsmode_do:
5192 }

```

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

32.3 Examples

sexample

```

5193 \keys_define:nn {stex / sexample }{
5194   type      .str_set_x:N = \exampletype,
5195   id        .str_set_x:N = \sexampleid,
5196   title     .tl_set:N     = \sexampletitle,
5197   name      .str_set_x:N = \sexamplename ,
5198   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5199 }
5200
5201 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5202   \str_clear:N \sexampletype
5203   \str_clear:N \sexampleid
5204   \str_clear:N \sexamplename
5205   \tl_clear:N \sexampletitle
5206   \clist_clear:N \l__stex_statements_sexample_for_clist
5207   \keys_set:nn { stex / sexample }{ #1 }
5208 }
5209
5210 \NewDocumentEnvironment{sexample}{0{}}{
5211   \__stex_statements_sexample_args:n{ #1 }
5212   \stex_reactivate_macro:N \premise
5213   \stex_reactivate_macro:N \conclusion
5214   \stex_if_smsmode:F {
5215     \seq_clear:N \l_tmpa_seq
5216     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5217       \tl_if_empty:nF{ ##1 }{
5218         \stex_get_symbol:n { ##1 }
5219         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5220           \l_stex_get_symbol_uri_str
5221         }
5222       }
5223     }
5224     \exp_args:Nnnx
5225     \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5226     \str_if_empty:NF \sexampletype {
5227       \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5228     }
5229     \str_if_empty:NF \sexamplename {
5230       \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5231     }
5232     \clist_set:N \l_tmpa_clist \sexampletype
5233     \tl_clear:N \l_tmpa_tl
5234     \clist_map_inline:Nn \l_tmpa_clist {
5235       \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5236         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5237       }
5238     }
5239     \tl_if_empty:NTF \l_tmpa_tl {
5240       \__stex_statements_sexample_start:
5241     }{
5242       \l_tmpa_tl
5243     }

```

```

5244 }
5245 \str_if_empty:NF \sexampleid {
5246   \stex_ref_new_doc_target:n \sexampleid
5247 }
5248 \stex_smsmode_do:
5249 ){
5250   \str_if_empty:NF \sexamplename {
5251     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5252   }
5253   \stex_if_smsmode:F {
5254     \clist_set:Nn \l_tmpa_clist \sexamplotype
5255     \tl_clear:N \l_tmpa_tl
5256     \clist_map_inline:Nn \l_tmpa_clist {
5257       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5258         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5259       }
5260     }
5261     \tl_if_empty:NTF \l_tmpa_tl {
5262       \__stex_statements_sexample_end:
5263     }{
5264       \l_tmpa_tl
5265     }
5266     \end{stex_annotate_env}
5267   }
5268 }

```

\stexpatchexample

```

5269
5270 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5271   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5272     (\sexamplotype)
5273   }~}
5274 }
5275 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5276
5277 \newcommand\stexpatchexample[3]{} {
5278   \str_set:Nx \l_tmpa_str{ #1 }
5279   \str_if_empty:NTF \l_tmpa_str {
5280     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5281     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5282   }{
5283     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5284     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5285   }
5286 }

```

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

\inlineex inline:

```

5287 \keys_define:nn {stex / inlineex }{
5288   type      .str_set_x:N = \sexamplotype,
5289   id        .str_set_x:N = \sexampleid,
5290   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5291   name      .str_set_x:N = \sexamplename

```

```

5292 }
5293 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5294   \str_clear:N \sexamplotype
5295   \str_clear:N \sexampleid
5296   \str_clear:N \sexamplename
5297   \clist_clear:N \l__stex_statements_sexample_for_clist
5298   \keys_set:nn { stex / inlineex }{ #1 }
5299 }
5300 \NewDocumentCommand \inlineex { 0{ } m } {
5301   \beginngroup
5302   \stex_reactivate_macro:N \premise
5303   \stex_reactivate_macro:N \conclusion
5304   \__stex_statements_inlineex_args:n{ #1 }
5305   \str_if_empty:NF \sexampleid {
5306     \stex_ref_new_doc_target:n \sexampleid
5307   }
5308   \stex_if_smsmode:TF{
5309     \str_if_empty:NF \sexamplename {
5310       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5311     }
5312   }{
5313     \seq_clear:N \l_tmpa_seq
5314     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5315       \tl_if_empty:nF{ ##1 }{
5316         \stex_get_symbol:n { ##1 }
5317         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5318           \l_stex_get_symbol_uri_str
5319         }
5320       }
5321     }
5322     \exp_args:Nnx
5323     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5324       \str_if_empty:NF \sexamplotype {
5325         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{}
5326       }
5327       #2
5328       \str_if_empty:NF \sexamplename {
5329         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5330         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5331       }
5332     }
5333   }
5334   \endgroup
5335   \stex_smsmode_do:
5336 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5337 \keys_define:nn { stex / sparagraph } {
5338   id          .str_set_x:N    = \sparagraphid ,

```

```

5339 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5340 type .str_set_x:N = \sparagraphtype ,
5341 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5342 from .tl_set:N = \sparagraphfrom ,
5343 to .tl_set:N = \sparagraphto ,
5344 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5345 name .str_set:N = \sparagraphname
5346 }
5347
5348 \cs_new_protected:Nn \stex_sparagraph_args:n {
5349 \tl_clear:N \l_stex_sparagraph_title_tl
5350 \tl_clear:N \sparagraphfrom
5351 \tl_clear:N \sparagraphto
5352 \tl_clear:N \l_stex_sparagraph_start_tl
5353 \str_clear:N \sparagraphid
5354 \str_clear:N \sparagraphtype
5355 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5356 \str_clear:N \sparagraphname
5357 \keys_set:nn { stex / sparagraph }{ #1 }
5358 }
5359 \newif\if@in@omtext\@in@omtextfalse
5360
5361 \NewDocumentEnvironment {sparagraph} { 0{} } {
5362 \stex_sparagraph_args:n { #1 }
5363 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5364 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5365 }{
5366 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5367 }
5368 \@in@omtexttrue
5369 \stex_if_smsmode:F {
5370 \seq_clear:N \l_tmpa_seq
5371 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5372 \tl_if_empty:NF{ ##1 }{
5373 \stex_get_symbol:n { ##1 }
5374 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5375 \l_stex_get_symbol_uri_str
5376 }
5377 }
5378 }
5379 \exp_args:Nnnx
5380 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5381 \str_if_empty:NF \sparagraphtype {
5382 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5383 }
5384 \str_if_empty:NF \sparagraphfrom {
5385 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5386 }
5387 \str_if_empty:NF \sparagraphto {
5388 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5389 }
5390 \str_if_empty:NF \sparagraphname {
5391 \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5392 }

```

```

5393 \clist_set:No \l_tmpa_clist \sparagraphtype
5394 \tl_clear:N \l_tmpa_tl
5395 \clist_map_inline:Nn \sparagraphtype {
5396   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5397     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5398   }
5399 }
5400 \tl_if_empty:NTF \l_tmpa_tl {
5401   \__stex_statements_sparagraph_start:
5402 }{
5403   \l_tmpa_tl
5404 }
5405 }
5406 \clist_set:No \l_tmpa_clist \sparagraphtype
5407 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}
5408 {
5409   \stex_reactivate_macro:N \definiendum
5410   \stex_reactivate_macro:N \definame
5411   \stex_reactivate_macro:N \Definame
5412   \stex_reactivate_macro:N \premise
5413   \stex_reactivate_macro:N \definien
5414 }
5415 \str_if_empty:NTF \sparagraphid {
5416   \str_if_empty:NTF \sparagraphname {
5417     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5418       \stex_ref_new_doc_target:n {}
5419     }
5420   } {
5421     \stex_ref_new_doc_target:n {}
5422   }
5423 } {
5424   \stex_ref_new_doc_target:n \sparagraphid
5425 }
5426 \exp_args:NNx
5427 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5428   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5429     \tl_if_empty:nF{ ##1 }{
5430       \stex_get_symbol:n { ##1 }
5431       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5432     }
5433   }
5434 }
5435 \stex_smsmode_do:
5436 \ignorespacesandpars
5437 }{
5438   \str_if_empty:NF \sparagraphname {
5439     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5440     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5441   }
5442   \stex_if_smsmode:F {
5443     \clist_set:No \l_tmpa_clist \sparagraphtype
5444     \tl_clear:N \l_tmpa_tl
5445     \clist_map_inline:Nn \l_tmpa_clist {
5446       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{

```

```

5447         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5448     }
5449 }
5450 \tl_if_empty:NTF \l_tmpa_tl {
5451     \__stex_statements_sparagraph_end:
5452 }{
5453     \l_tmpa_tl
5454 }
5455 \end{stex_annotate_env}
5456 }
5457 }

```

\stexpatchparagraph

```

5458
5459 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5460     \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5461         \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5462             \titleemph{\l_stex_sparagraph_title_tl}:~
5463         }
5464     }{
5465         \titleemph{\l_stex_sparagraph_start_tl}~
5466     }
5467 }
5468 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5469
5470 \newcommand\stexpatchparagraph[3] [] {
5471     \str_set:Nx \l_tmpa_str{ #1 }
5472     \str_if_empty:NTF \l_tmpa_str {
5473         \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5474         \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5475     }{
5476         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5477         \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5478     }
5479 }
5480
5481 \keys_define:nn { stex / inlinepara } {
5482     id .str_set_x:N = \sparagraphid ,
5483     type .str_set_x:N = \sparagraphtype ,
5484     for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5485     from .tl_set:N = \sparagraphfrom ,
5486     to .tl_set:N = \sparagraphto ,
5487     name .str_set:N = \sparagraphname
5488 }
5489 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5490     \tl_clear:N \sparagraphfrom
5491     \tl_clear:N \sparagraphto
5492     \str_clear:N \sparagraphid
5493     \str_clear:N \sparagraphtype
5494     \clist_clear:N \l__stex_statements_sparagraph_for_clist
5495     \str_clear:N \sparagraphname
5496     \keys_set:nn { stex / inlinepara }{ #1 }
5497 }
5498 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

5499 \begingroup
5500 \__stex_statements_inlinepara_args:n{ #1 }
5501 \clist_set:No \l_tmpa_clist \sparagraphtype
5502 \str_if_empty:NTF \sparagraphid {
5503   \str_if_empty:NTF \sparagraphname {
5504     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5505       \stex_ref_new_doc_target:n {}
5506     }
5507   } {
5508     \stex_ref_new_doc_target:n {}
5509   }
5510 } {
5511   \stex_ref_new_doc_target:n \sparagraphid
5512 }
5513 \stex_if_smsmode:TF{
5514   \str_if_empty:NF \sparagraphname {
5515     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5516   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5517 }
5518 }{
5519   \seq_clear:N \l_tmpa_seq
5520   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5521     \tl_if_empty:nF{ ##1 }{
5522       \stex_get_symbol:n { ##1 }
5523       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5524         \l_stex_get_symbol_uri_str
5525       }
5526     }
5527   }
5528   \exp_args:Nnx
5529   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5530     \str_if_empty:NF \sparagraphtype {
5531       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5532     }
5533     \str_if_empty:NF \sparagraphfrom {
5534       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5535     }
5536     \str_if_empty:NF \sparagraphto {
5537       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5538     }
5539     \str_if_empty:NF \sparagraphname {
5540       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5541     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5542     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5543   }
5544   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5545     \clist_map_inline:Nn \l_tmpa_seq {
5546       \stex_ref_new_sym_target:n {##1}
5547     }
5548   }
5549   #2
5550 }
5551 }
5552 \endgroup

```



```

5553 \stex_smsmode_do:
5554 }
5555
5556 (End definition for \stexpatchparagraph. This function is documented on page 42.)
5557 \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

```
5557 <*package>
5558 <@@=stex_sproof>
5559
5560 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5561
```

33.2 Proofs

We first define some keys for the proof environment.

```
5562 \keys_define:nn { stex / spf } {
5563   id          .str_set_x:N = \spfid,
5564   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5565   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5566   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5567   type        .str_set_x:N = \spftype,
5568   title       .tl_set:N    = \spftitle,
5569   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5570   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5571   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5572 }
5573 \cs_new_protected:Nn \l__stex_sproof_spf_args:n {
5574   \str_clear:N \spfid
5575   \tl_clear:N \l__stex_sproof_spf_for_tl
5576   \tl_clear:N \l__stex_sproof_spf_from_tl
5577   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5578   \str_clear:N \spftype
5579   \tl_clear:N \spftitle
5580   \tl_clear:N \l__stex_sproof_spf_continues_tl
5581   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

⁸EdNOTE: need an implementation for L^AT_EX_ML

```

5582 \tl_clear:N \l__stex_sproof_spf_method_tl
5583 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5584 \keys_set:nn { stex / spf }{ #1 }
5585 }

```

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

```

5586 \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.

```

5587 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5588 \cs_new_protected:Npn \sproofnumber {
5589   \int_set:Nn \l_tmpa_int {1}
5590   \bool_while_do:nn {
5591     \int_compare_p:nNn {
5592       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5593     } > 0
5594   }{
5595     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5596     \int_incr:N \l_tmpa_int
5597   }
5598 }
5599 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5600   \int_set:Nn \l_tmpa_int {1}
5601   \bool_while_do:nn {
5602     \int_compare_p:nNn {
5603       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5604     } > 0
5605   }{
5606     \int_incr:N \l_tmpa_int
5607   }
5608   \int_compare:nNnF \l_tmpa_int = 1 {
5609     \int_decr:N \l_tmpa_int
5610   }
5611   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5612     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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

```

5613 }
5614 }
5615
5616 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5617   \int_set:Nn \l_tmpa_int {1}
5618   \bool_while_do:nn {
5619     \int_compare_p:nNn {
5620       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5621     } > 0
5622   }{
5623     \int_incr:N \l_tmpa_int
5624   }
5625   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5626 }
5627
5628 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5629   \int_set:Nn \l_tmpa_int {1}
5630   \bool_while_do:nn {
5631     \int_compare_p:nNn {
5632       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5633     } > 0
5634   }{
5635     \int_incr:N \l_tmpa_int
5636   }
5637   \int_decr:N \l_tmpa_int
5638   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5639 }

```

\sproofend This macro places a little box at the end of the line if there is space, or at the end of the next line if there isn't

```

5640 \def\sproof@box{
5641   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5642 }
5643 \def\sproofend{
5644   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5645     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5646   }
5647 }

```

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

spf@*@kw

```

5648 \def\spf@proofsketch@kw{Proof~Sketch}
5649 \def\spf@proof@kw{Proof}
5650 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5651 \AddToHook{begindocument}{
5652   \ltx@ifpackageloaded{babel}{
5653     \makeatletter
5654     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5655     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5656       \input{sproof-ngerman.ldf}

```

```

5657     }
5658     \clist_if_in:NnT \l_tmpa_clist {finnish}{
5659       \input{sproof-finnish.ldf}
5660     }
5661     \clist_if_in:NnT \l_tmpa_clist {french}{
5662       \input{sproof-french.ldf}
5663     }
5664     \clist_if_in:NnT \l_tmpa_clist {russian}{
5665       \input{sproof-russian.ldf}
5666     }
5667     \makeatother
5668   }{}
5669 }

```

spfsketch

```

5670 \newcommand\spfsketch[2] [] {
5671   \beginingroup
5672   \let \premise \stex_proof_premise:
5673   \__stex_sproof_spf_args:n{#1}
5674   \stex_if_smsmode:TF {
5675     \str_if_empty:NF \spfid {
5676       \stex_ref_new_doc_target:n \spfid
5677     }
5678   }{
5679     \seq_clear:N \l_tmpa_seq
5680     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5681       \tl_if_empty:nF{ ##1 }{
5682         \stex_get_symbol:n { ##1 }
5683         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5684           \l_stex_get_symbol_uri_str
5685         }
5686       }
5687     }
5688     \exp_args:Nnx
5689     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5690       \str_if_empty:NF \spftype {
5691         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5692       }
5693       \clist_set:No \l_tmpa_clist \spftype
5694       \tl_set:Nn \l_tmpa_tl {
5695         \titleemph{
5696           \tl_if_empty:NTF \spftitle {
5697             \spf@proofsketch@kw
5698           }{
5699             \spftitle
5700           }
5701         }::~
5702       }
5703       \clist_map_inline:Nn \l_tmpa_clist {
5704         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5705           \tl_clear:N \l_tmpa_tl
5706         }
5707       }
5708       \str_if_empty:NF \spfid {

```

```

5709         \stex_ref_new_doc_target:n \spfid
5710     }
5711     \l_tmpa_tl #2 \sproofend
5712 }
5713 }
5714 \endgroup
5715 \stex_smsmode_do:
5716 }
5717

```

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

EdN:9
EdN:10 **spfeq** This is very similar to \spfsketch, but uses a computation array⁹¹⁰

```

5718 \newenvironment{spfeq}[2][]{
5719   \__stex_sproof_spf_args:n{#1}
5720   \let \premise \stex_proof_premise:
5721   \stex_if_smsmode:TF {
5722     \str_if_empty:NF \spfid {
5723       \stex_ref_new_doc_target:n \spfid
5724     }
5725   }{
5726     \seq_clear:N \l_tmpa_seq
5727     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5728       \tl_if_empty:NF{ ##1 }{
5729         \stex_get_symbol:n { ##1 }
5730         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5731           \l_stex_get_symbol_uri_str
5732         }
5733       }
5734     }
5735     \exp_args:Nnnx
5736     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5737     \str_if_empty:NF \spftype {
5738       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5739     }
5740
5741     \clist_set:No \l_tmpa_clist \spftype
5742     \tl_clear:N \l_tmpa_tl
5743     \clist_map_inline:Nn \l_tmpa_clist {
5744       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5745         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5746       }
5747       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5748         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5749       }
5750     }
5751     \tl_if_empty:NTF \l_tmpa_tl {
5752       \__stex_sproof_spfeq_start:
5753     }{
5754       \l_tmpa_tl
5755     }{-#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

```

5756 \str_if_empty:NF \spfid {
5757 \stex_ref_new_doc_target:n \spfid
5758 }
5759 \begin{displaymath}\begin{array}{rc1l}
5760 }
5761 \stex_smsmode_do:
5762 }{
5763 \stex_if_smsmode:F {
5764 \end{array}\end{displaymath}
5765 \clist_set:No \l_tmpa_clist \spftype
5766 \tl_clear:N \l_tmpa_tl
5767 \clist_map_inline:Nn \l_tmpa_clist {
5768 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5769 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5770 }
5771 }
5772 \tl_if_empty:NTF \l_tmpa_tl {
5773 \__stex_sproof_spfeq_end:
5774 }{
5775 \l_tmpa_tl
5776 }
5777 \end{stex_annotate_env}
5778 }
5779 }
5780
5781 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5782 \titleemph{
5783 \tl_if_empty:NTF \spftitle {
5784 \spf@proof@kw
5785 }{
5786 \spftitle
5787 }
5788 }:
5789 }
5790 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5791
5792 \newcommand\stexpatchspfeq[3] [] {
5793 \str_set:Nx \l_tmpa_str{ #1 }
5794 \str_if_empty:NTF \l_tmpa_str {
5795 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5796 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5797 }{
5798 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5799 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5800 }
5801 }
5802

```

(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.

```

5803 \newenvironment{sproof}[2] []{

```

```

5804 \let \premise \stex_proof_premise:
5805 \intarray_gzero:N \l__stex_sproof_counter_intarray
5806 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5807 \__stex_sproof_spf_args:n{#1}
5808 \stex_if_smsmode:TF {
5809   \str_if_empty:NF \spfid {
5810     \stex_ref_new_doc_target:n \spfid
5811   }
5812 }{
5813   \seq_clear:N \l_tmpa_seq
5814   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5815     \tl_if_empty:NF{ ##1 }{
5816       \stex_get_symbol:n { ##1 }
5817       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5818         \l_stex_get_symbol_uri_str
5819       }
5820     }
5821   }
5822   \exp_args:Nnnx
5823   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {},}}
5824   \str_if_empty:NF \spftype {
5825     \stex_annotate_invisible:nnn{type}{\spftype}{}
5826   }
5827
5828   \clist_set:No \l_tmpa_clist \spftype
5829   \tl_clear:N \l_tmpa_tl
5830   \clist_map_inline:Nn \l_tmpa_clist {
5831     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5832       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5833     }
5834     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5835       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5836     }
5837   }
5838   \tl_if_empty:NTF \l_tmpa_tl {
5839     \__stex_sproof_sproof_start:
5840   }{
5841     \l_tmpa_tl
5842   }{~#2}
5843   \str_if_empty:NF \spfid {
5844     \stex_ref_new_doc_target:n \spfid
5845   }
5846   \begin{description}
5847 }
5848 \stex_smsmode_do:
5849 }{
5850 \stex_if_smsmode:F{
5851   \end{description}
5852   \clist_set:No \l_tmpa_clist \spftype
5853   \tl_clear:N \l_tmpa_tl
5854   \clist_map_inline:Nn \l_tmpa_clist {
5855     \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5856       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5857     }

```



```

5858     }
5859     \tl_if_empty:NTF \l_tmpa_tl {
5860       \__stex_sproof_sproof_end:
5861     }{
5862       \l_tmpa_tl
5863     }
5864     \end{stex_annotate_env}
5865   }
5866 }
5867
5868 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5869   \par\noindent\titleemph{
5870     \tl_if_empty:NTF \spftype {
5871       \spf@proof@kw
5872     }{
5873       \spftype
5874     }
5875   }:
5876 }
5877 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5878
5879 \newcommand\stexpatchproof[3] [] {
5880   \str_set:Nx \l_tmpa_str{ #1 }
5881   \str_if_empty:NTF \l_tmpa_str {
5882     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5883     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5884   }{
5885     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5886     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5887   }
5888 }

```

\spfidea

```

5889 \newcommand\spfidea[2] []{
5890   \__stex_sproof_spf_args:n{#1}
5891   \titleemph{
5892     \tl_if_empty:NTF \spftype {Proof~Idea}{
5893       \spftype
5894     }:
5895   }~#2
5896   \sproofend
5897 }

```

(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

```

5898 \newenvironment{spfstep}[1] []{
5899   \__stex_sproof_spf_args:n{#1}
5900   \stex_if_smsmode:TF {

```

```

5901 \str_if_empty:NF \spfid {
5902   \stex_ref_new_doc_target:n \spfid
5903 }
5904 }{
5905   \@in@omtexttrue
5906   \seq_clear:N \l_tmpa_seq
5907   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5908     \tl_if_empty:nF{ ##1 }{
5909       \stex_get_symbol:n { ##1 }
5910       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5911         \l_stex_get_symbol_uri_str
5912       }
5913     }
5914   }
5915   \exp_args:Nnnx
5916   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5917   \str_if_empty:NF \spftype {
5918     \stex_annotate_invisible:nnn{type}{\spftype}{}
5919   }
5920   \clist_set:No \l_tmpa_clist \spftype
5921   \tl_set:Nn \l_tmpa_tl {
5922     \item[\sproofnumber]
5923     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5924   }
5925   \clist_map_inline:Nn \l_tmpa_clist {
5926     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5927       \tl_clear:N \l_tmpa_tl
5928     }
5929   }
5930   \l_tmpa_tl
5931   \tl_if_empty:NF \spftitle {
5932     {(\titleemph{\spftitle})\enspace}
5933   }
5934   \str_if_empty:NF \spfid {
5935     \stex_ref_new_doc_target:n \spfid
5936   }
5937 }
5938 \stex_smsmode_do:
5939 \ignorespacesandpars
5940 }{
5941   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5942     \__stex_sproof_inc_counter:
5943   }
5944   \stex_if_smsmode:F {
5945     \end{stex_annotate_env}
5946   }
5947 }

```

sproofcomment

```

5948 \newenvironment{sproofcomment}[1][]{
5949   \__stex_sproof_spf_args:n{#1}
5950   \clist_set:No \l_tmpa_clist \spftype
5951   \tl_set:Nn \l_tmpa_tl {
5952     \item[\sproofnumber]

```

```

5953 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5954 }
5955 \clist_map_inline:Nn \l_tmpa_clist {
5956 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5957 \tl_clear:N \l_tmpa_tl
5958 }
5959 }
5960 \l_tmpa_tl
5961 }{
5962 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5963 \__stex_sproof_inc_counter:
5964 }
5965 }

```

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.

```

5966 \newenvironment{subproof}[2][]{
5967 \__stex_sproof_spf_args:n{#1}
5968 \stex_if_smsmode:TF{
5969 \str_if_empty:NF \spfid {
5970 \stex_ref_new_doc_target:n \spfid
5971 }
5972 }{
5973 \seq_clear:N \l_tmpa_seq
5974 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5975 \tl_if_empty:nF{ ##1 }{
5976 \stex_get_symbol:n { ##1 }
5977 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5978 \l_stex_get_symbol_uri_str
5979 }
5980 }
5981 }
5982 \exp_args:Nnnx
5983 \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5984 \str_if_empty:NF \spftype {
5985 \stex_annotate_invisible:nnn{type}{\spftype}{}}
5986 }
5987
5988 \clist_set:No \l_tmpa_clist \spftype
5989 \tl_set:Nn \l_tmpa_tl {
5990 \item[\sproofnumber]
5991 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5992 }
5993 \clist_map_inline:Nn \l_tmpa_clist {
5994 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5995 \tl_clear:N \l_tmpa_tl
5996 }
5997 }
5998 \l_tmpa_tl
5999 \tl_if_empty:NF \spftitle {
6000 {(\titleemph{\spftitle})\enspace}
6001 }

```

```

6002     {~#2}
6003     \str_if_empty:NF \spfid {
6004         \stex_ref_new_doc_target:n \spfid
6005     }
6006 }
6007 \__stex_sproof_add_counter:
6008 \stex_smsmode_do:
6009 }{
6010     \__stex_sproof_remove_counter:
6011     \bool_if:NT \l__stex_sproof_inc_counter_bool {
6012         \__stex_sproof_inc_counter:
6013     }
6014     \stex_if_smsmode:F{
6015         \end{stex_annotate_env}
6016     }
6017 }

```

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

```

6018 \newenvironment{spfcases}[2][]{
6019     \tl_if_empty:nTF{#1}{
6020         \begin{subproof}[method=by-cases]{#2}
6021     }{
6022         \begin{subproof}[#1,method=by-cases]{#2}
6023     }
6024 }{
6025     \end{subproof}
6026 }

```

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

```

6027 \newenvironment{spfcase}[2][]{
6028     \__stex_sproof_spf_args:n{#1}
6029     \stex_if_smsmode:TF {
6030         \str_if_empty:NF \spfid {
6031             \stex_ref_new_doc_target:n \spfid
6032         }
6033     }{
6034         \seq_clear:N \l_tmpa_seq
6035         \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
6036             \tl_if_empty:nF{ ##1 }{
6037                 \stex_get_symbol:n { ##1 }
6038                 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
6039                     \l_stex_get_symbol_uri_str
6040                 }
6041             }
6042         }
6043         \exp_args:Nnnx
6044         \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
6045         \str_if_empty:NF \spftype {
6046             \stex_annotate_invisible:nnn{type}{\spftype}{}}
6047     }
6048     \clist_set:Nn \l_tmpa_clist \spftype
6049     \tl_set:Nn \l_tmpa_tl {
6050         \item[\sproofnumber]

```

```

6051     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6052   }
6053   \clist_map_inline:Nn \l_tmpa_clist {
6054     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6055       \tl_clear:N \l_tmpa_tl
6056     }
6057   }
6058   \l_tmpa_tl
6059   \tl_if_empty:nF{#2}{
6060     \titleemph{#2}:~
6061   }
6062 }
6063 \__stex_sproof_add_counter:
6064 \stex_smsmode_do:
6065 ){
6066   \__stex_sproof_remove_counter:
6067   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6068     \__stex_sproof_inc_counter:
6069   }
6070   \stex_if_smsmode:F{
6071     \clist_set:No \l_tmpa_clist \spftype
6072     \tl_set:Nn \l_tmpa_tl{\sproofend}
6073     \clist_map_inline:Nn \l_tmpa_clist {
6074       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6075         \tl_clear:N \l_tmpa_tl
6076       }
6077     }
6078     \l_tmpa_tl
6079     \end{stex_annotate_env}
6080   }
6081 }

```

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

```

6082 \newcommand\spfcasesketch[3][]{
6083   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6084 }

```

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.

```

6085 \keys_define:nn { stex / just }{
6086   id      .str_set:x:N = \l__stex_sproof_just_id_str,
6087   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
6088   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
6089   args     .tl_set:N    = \l__stex_sproof_just_args_tl
6090 }

```

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

```
6091 \newenvironment{justification}[1] [] {}{}
```

\premise

```
6092 \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.

```
6093 \newcommand\justarg[2] [] {#2}
```

```
6094 \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

```
6095 <*package>
6096
6097 %%%%%%%%%% others.dtx %%%%%%%%%%
6098
6099 <@@=stex_others>
        Warnings and error messages
6100 % None

\MSC Math subject classifier

6101 \NewDocumentCommand \MSC {m} {
6102 % TODO
6103 }

(End definition for \MSC. This function is documented on page ??.)
        Patching tikzinput, if loaded
6104 \@ifpackageloaded{tikzinput}{
6105 \RequirePackage{stex-tikzinput}
6106 }{}
6107 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6108 <*package>
6109 <@@=stex_modules>
6110
6111 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6112
6113 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
6114 \begingroup
6115 \stex_module_setup:nn{
6116   ns=\c_stex_metatheory_ns_str,
6117   meta=NONE
6118 }{Metatheory}
6119 \stex_reactivate_macro:N \symdecl
6120 \stex_reactivate_macro:N \notation
6121 \stex_reactivate_macro:N \symdef
6122 \ExplSyntaxOff
6123 \csname stex_suppress_html:n\endcsname{
6124   % is-a (a:A, a \in A, a is an A, etc.)
6125   \symdecl{isa}[args=ai]
6126   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6127   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6128   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6129
6130   % bind (\forall, \Pi, \lambda etc.)
6131   \symdecl{bind}[args=Bi]
6132   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
6133   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6134   \notation{bind}[depfun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
6135
6136   % implicit bind
6137   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6138
6139   % dummy variable
6140   \symdecl{dummyvar}
6141   \notation{dummyvar}[underscore]{\comp\_}
6142   \notation{dummyvar}[dot]{\comp\cdot}
```



```

6143 \notation{dummyvar}[dash]{\comp{\rm --}}
6144
6145 %fromto (function space, Hom-set, implication etc.)
6146 \symdecl{fromto}[args=ai]
6147 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6148 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6149
6150 % mapto (lambda etc.)
6151 \symdecl{mapto}[args=Bi]
6152 %\notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6153 %\notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6154 %\notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6155
6156 % function/operator application
6157 \symdecl{apply}[args=ia]
6158 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6159 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6160
6161 % collection of propositions/booleans/truth values
6162 \symdecl{prop}[name=proposition]
6163 \notation{prop}[prop]{\comp{\rm prop}}
6164 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6165
6166 \symdecl{judgmentholds}[args=1]
6167 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6168
6169 % sequences
6170 \symdecl{seqtype}[args=1]
6171 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6172
6173 \symdecl{seqexpr}[args=a]
6174 \notation{seqexpr}[angle,prec=nobrackets]{\comp\angle #1\comp\rangle}{##1\comp,##2}
6175
6176 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6177 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6178
6179 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6180 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6181 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}}{##1\comp,##2}
6182
6183 % letin (''let'', local definitions, variable substitution)
6184 \symdecl{letin}[args=bii]
6185 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\;#2\; \comp{\rm in}}\;#3}
6186 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6187 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6188
6189 % structures
6190 \symdecl*{module-type}[args=1]
6191 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6192 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6193 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6194
6195 % objects
6196 \symdecl{object}

```

```

6197 \notation{object}{\comp{\mathtt{OBJECT}}}
6198
6199 }
6200 \ExplSyntaxOn
6201 \stex_add_to_current_module:n{
6202   \let\nappa\apply
6203   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6204   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6205   \def\livar{\csname sequence-index\endcsname[li]}
6206   \def\uivar{\csname sequence-index\endcsname[ui]}
6207   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6208   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6209   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6210 }
6211 \__stex_modules_end_module:
6212 \endgroup
6213 \</package>

```

Chapter 36

Tikzinput Implementation

```
6214 <*package>
6215
6216 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6217
6218 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6219 \RequirePackage{l3keys2e}
6220
6221 \keys_define:nn { tikzinput } {
6222   image .bool_set:N = \c_tikzinput_image_bool,
6223   image .default:n = false ,
6224   unknown .code:n = {}
6225 }
6226
6227 \ProcessKeysOptions { tikzinput }
6228
6229 \bool_if:NTF \c_tikzinput_image_bool {
6230   \RequirePackage{graphicx}
6231
6232   \providecommand\usetikzlibrary[]{}
6233   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6234 }{
6235   \RequirePackage{tikz}
6236   \RequirePackage{standalone}
6237
6238   \newcommand \tikzinput [2] [] {
6239     \setkeys{Gin}{#1}
6240     \ifx \Gin@ewidth \Gin@exclamation
6241       \ifx \Gin@eheight \Gin@exclamation
6242         \input { #2 }
6243       \else
6244         \resizebox{!}{ \Gin@eheight }{
6245           \input { #2 }
6246         }
6247       \fi
6248     \else
6249       \ifx \Gin@eheight \Gin@exclamation
6250         \resizebox{ \Gin@ewidth }{!}{
6251           \input { #2 }
```

```

6252     }
6253     \else
6254         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6255             \input { #2 }
6256         }
6257     \fi
6258 \fi
6259 }
6260 }
6261
6262 \newcommand \ctikzinput [2] [] {
6263     \begin{center}
6264         \tikzinput [1] {#2}
6265     \end{center}
6266 }
6267
6268 \@ifpackageloaded{stex}{
6269     \RequirePackage{stex-tikzinput}
6270 }{}
6271
6272 </package>
6273 <*stex>
6274 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6275 \RequirePackage{stex}
6276 \RequirePackage{tikzinput}
6277
6278 \newcommand\mhtikzinput [2] [] {%
6279     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6280     \stex_in_repository:nn\Gin@mhrepos{
6281         \tikzinput [1]{\mhpath{##1}{#2}}
6282     }
6283 }
6284 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
6285 </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.

```
6286 \*cls)
6287 \@@=document_structure)
6288 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6289 \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`

```
6290 \keys_define:nn{ document-structure / pkg }{
6291   class      .str_set_x:N = \c_document_structure_class_str,
6292   minimal    .bool_set:N = \c_document_structure_minimal_bool,
6293   report     .code:n      = {
6294     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6295     \str_set:Nn \c_document_structure_class_str {report}
6296   },
6297   book       .code:n      = {
6298     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6299     \str_set:Nn \c_document_structure_class_str {book}
6300   },
6301   bookpart   .code:n      = {
6302     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6303     \str_set:Nn \c_document_structure_class_str {book}
6304     \str_set:Nn \c_document_structure_topsect_str {chapter}
6305   },
```

```

6306 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6307 unknown     .code:n      = {
6308   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6309 }
6310 }
6311 \ProcessKeysOptions{ document-structure / pkg }
6312 \str_if_empty:NT \c_document_structure_class_str {
6313   \str_set:Nn \c_document_structure_class_str {article}
6314 }
6315 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6316   {\c_document_structure_class_str}
6317

```

37.3 Beefing up the document environment

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

```

6318 \RequirePackage{document-structure}
6319 \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

```

6320 \keys_define:nn { document-structure / document }{
6321   id .str_set_x:N = \c_document_structure_document_id_str
6322 }
6323 \let\__document_structure_orig_document=\document
6324 \renewcommand{\document}[1][]{
6325   \keys_set:nn{ document-structure / document }{ #1 }
6326   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6327   \__document_structure_orig_document
6328 }

```

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

```

6329 }
6330 \</cls>

```

37.4 Implementation: document-structure Package

```

6331 \<package>
6332 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6333 \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

```

6334
6335 \keys_define:nn{ document-structure / pkg }{
6336   class      .str_set_x:N = \c_document_structure_class_str,
6337   topsect     .str_set_x:N = \c_document_structure_topsect_str,
6338   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6339 }
6340 \ProcessKeysOptions{ document-structure / pkg }
6341 \str_if_empty:NT \c_document_structure_class_str {
6342   \str_set:Nn \c_document_structure_class_str {article}
6343 }
6344 \str_if_empty:NT \c_document_structure_topsect_str {
6345   \str_set:Nn \c_document_structure_topsect_str {section}
6346 }

```

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

```

6347 \RequirePackage{xspace}
6348 \RequirePackage{comment}
6349 \AddToHook{begindocument}{
6350   \ltx@ifpackageloaded{babel}{
6351     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6352     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6353       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6354     }
6355   }{}
6356 }

```

`\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.

```

6357 \int_new:N \l_document_structure_section_level_int
6358 \str_case:VnF \c_document_structure_topsect_str {
6359   {part}}{
6360     \int_set:Nn \l_document_structure_section_level_int {0}
6361   }
6362   {chapter}}{
6363     \int_set:Nn \l_document_structure_section_level_int {1}
6364   }
6365 }{
6366   \str_case:VnF \c_document_structure_class_str {
6367     {book}}{
6368       \int_set:Nn \l_document_structure_section_level_int {0}
6369     }
6370     {report}}{
6371       \int_set:Nn \l_document_structure_section_level_int {0}
6372     }
6373   }{
6374     \int_set:Nn \l_document_structure_section_level_int {2}
6375   }
6376 }

```

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

```
6377 \def\current@section@level{document}%
6378 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6379 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6380 \cs_new_protected:Npn \skipomgroup {
6381   \ifcase\l_document_structure_section_level_int
6382   \or\stepcounter{part}
6383   \or\stepcounter{chapter}
6384   \or\stepcounter{section}
6385   \or\stepcounter{subsection}
6386   \or\stepcounter{subsubsection}
6387   \or\stepcounter{paragraph}
6388   \or\stepcounter{subparagraph}
6389   \fi
6390 }
```

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

`blindfragment`

```
6391 \newcommand\at@begin@blindomgroup[1]{%
6392 \newenvironment{blindfragment}
6393 {
6394   \int_incr:N\l_document_structure_section_level_int
6395   \at@begin@blindomgroup\l_document_structure_section_level_int
6396 }{}}
```

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

```
6397 \newcommand\omgroup@nonum[2]{
6398   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6399   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6400 }
```

(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.

```
6401 \newcommand\omgroup@num[2]{
```

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


```

6402 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6403   \@nameuse{#1}{#2}
6404 }{
6405   \cs_if_exist:NTF\rdfmata@sectioning{
6406     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6407   }{
6408     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6409   }
6410 }
6411 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6412 }

```

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

sfragment

```

6413 \keys_define:nn { document-structure / omgroup }{
6414   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6415   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6416   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6417   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6418   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6419   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6420   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6421   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6422   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6423   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6424 }
6425 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6426   \str_clear:N \l__document_structure_omgroup_id_str
6427   \str_clear:N \l__document_structure_omgroup_date_str
6428   \clist_clear:N \l__document_structure_omgroup_creators_clist
6429   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6430   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6431   \tl_clear:N \l__document_structure_omgroup_type_tl
6432   \tl_clear:N \l__document_structure_omgroup_short_tl
6433   \tl_clear:N \l__document_structure_omgroup_display_tl
6434   \tl_clear:N \l__document_structure_omgroup_intro_tl
6435   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6436   \keys_set:nn { document-structure / omgroup } { #1 }
6437 }

```

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.

```

6438 \newif\if@mainmatter\@mainmattertrue
6439 \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.

```

6440 \keys_define:nn { document-structure / sectioning }{
6441   name .str_set_x:N = \l__document_structure_sect_name_str ,
6442   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6443   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6444   clear .default:n = {true} ,
6445   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6446   num      .default:n      = {true}
6447 }
6448 \cs_new_protected:Nn \__document_structure_sect_args:n {
6449   \str_clear:N \l__document_structure_sect_name_str
6450   \str_clear:N \l__document_structure_sect_ref_str
6451   \bool_set_false:N \l__document_structure_sect_clear_bool
6452   \bool_set_false:N \l__document_structure_sect_num_bool
6453   \keys_set:nn { document-structure / sectioning } { #1 }
6454 }
6455 \newcommand\omdoc@sectioning[3][]{
6456   \__document_structure_sect_args:n {#1}
6457   \let\omdoc@sect@name\l__document_structure_sect_name_str
6458   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6459   \if@mainmatter% numbering not overridden by frontmatter, etc.
6460     \bool_if:NTF \l__document_structure_sect_num_bool {
6461       \omgroup@num{#2}{#3}
6462     }{
6463       \omgroup@nonum{#2}{#3}
6464     }
6465     \def\current@section@level{\omdoc@sect@name}
6466   \else
6467     \omgroup@nonum{#2}{#3}
6468   \fi
6469 }% 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.

```

6470 \newcommand\omgroup@redefine@addtocontents[1]{%
6471 %\edef\__document_structureimport{#1}%
6472 %\@for\@I:=\__document_structureimport\do{%
6473 %\edef\@path{\csname module@\@I @path\endcsname}%
6474 %\@ifundefined{tf@toc}\relax%
6475 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6476 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6477 %\def\addcontentsline##1##2##3{%
6478 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6479 %\else% hyperref.sty not loaded
6480 %\def\addcontentsline##1##2##3{%
6481 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6482 %\fi
6483 }% 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.

```

6484 \newenvironment{sfragment}[2][]{% keys, title
6485 {
6486   \__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.

```

6487 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6488   \omgroup@redefine@addtocontents{
6489     \@ifundefined{module@id}\used@modules%

```

```

6490     %{\@ifundefined{module@module@id @path}{\used@modules}\module@id}
6491   }
6492 }

```

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

```

6493 \int_incr:N\l_document_structure_section_level_int
6494 \ifcase\l_document_structure_section_level_int
6495   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6496   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6497   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6498   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6499   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6500   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6501   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6502 \fi
6503 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6504 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6505   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6506 }
6507 }% for customization
6508 {}

```

and finally, we localize the sections

```

6509 \newcommand\omdoc@part@kw{Part}
6510 \newcommand\omdoc@chapter@kw{Chapter}
6511 \newcommand\omdoc@section@kw{Section}
6512 \newcommand\omdoc@subsection@kw{Subsection}
6513 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6514 \newcommand\omdoc@paragraph@kw{paragraph}
6515 \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`

```

6516 \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).

```

6517 \cs_if_exist:NTF\frontmatter{
6518   \let\__document_structure_orig_frontmatter\frontmatter
6519   \let\frontmatter\relax
6520 }{
6521   \tl_set:Nn\__document_structure_orig_frontmatter{
6522     \clearpage
6523     \@mainmatterfalse
6524     \pagenumbering{roman}

```

```

6525 }
6526 }
6527 \cs_if_exist:NTF\backmatter{
6528   \let\__document_structure_orig_backmatter\backmatter
6529   \let\backmatter\relax
6530 }{
6531   \tl_set:Nn\__document_structure_orig_backmatter{
6532     \clearpage
6533     \@mainmatterfalse
6534     \pagenumbering{roman}
6535   }
6536 }

```

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.

```

6537 \newenvironment{frontmatter}{
6538   \__document_structure_orig_frontmatter
6539 }{
6540   \cs_if_exist:NTF\mainmatter{
6541     \mainmatter
6542   }{
6543     \clearpage
6544     \@mainmattertrue
6545     \pagenumbering{arabic}
6546   }
6547 }

```

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

```

6548 \newenvironment{backmatter}{
6549   \__document_structure_orig_backmatter
6550 }{
6551   \cs_if_exist:NTF\mainmatter{
6552     \mainmatter
6553   }{
6554     \clearpage
6555     \@mainmattertrue
6556     \pagenumbering{arabic}
6557   }
6558 }

```

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

```

6559 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6560 \def \c__document_structure_document_str{document}
6561 \newcommand\afterprematurestop{}
6562 \def\prematurestop@endomgroup{
6563   \unless\ifx\@currenvir\c__document_structure_document_str
6564     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
6565     \expandafter\prematurestop@endomgroup

```

```

6566 \fi
6567 }
6568 \providecommand\prematurestop{
6569 \message{Stopping~sTeX~processing~prematurely}
6570 \prematurestop@endomgroup
6571 \afterprematurestop
6572 \end{document}
6573 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

6574 \RequirePackage{etoolbox}
6575 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6576 \newrobustcmd\useSGvar[1]{%
6577 \@ifundefined{sTeX@Gvar@#1}
6578 {\PackageError{document-structure}
6579 {The sTeX Global variable #1 is undefined}
6580 {set it with \protect\setSGvar}}
6581 \@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.

```

6582 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6583 \@ifundefined{sTeX@Gvar@#1}
6584 {\PackageError{document-structure}
6585 {The sTeX Global variable #1 is undefined}
6586 {set it with \protect\setSGvar}}
6587 {\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.

```
6588 \*cls)
6589 \@@=notesslides)
6590 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6591 \RequirePackage{13keys2e}
6592
6593 \keys_define:nn{notesslides / cls}{
6594   class .code:n = {
6595     \PassOptionsToClass{\CurrentOption}{document-structure}
6596     \str_if_eq:nnT{#1}{book}{
6597       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6598     }
6599     \str_if_eq:nnT{#1}{report}{
6600       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6601     }
6602   },
6603   notes .bool_set:N = \c__notesslides_notes_bool ,
6604   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6605   unknown .code:n = {
6606     \PassOptionsToClass{\CurrentOption}{document-structure}
6607     \PassOptionsToClass{\CurrentOption}{beamer}
6608     \PassOptionsToPackage{\CurrentOption}{notesslides}
6609   }
6610 }
6611 \ProcessKeysOptions{ notesslides / cls }
6612 \bool_if:NTF \c__notesslides_notes_bool {
6613   \PassOptionsToPackage{notes=true}{notesslides}
6614 }{
6615   \PassOptionsToPackage{notes=false}{notesslides}
6616 }
6617 \</cls)
```

now we do the same for the notesslides package.

```

6618 <*package>
6619 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6620 \RequirePackage{13keys2e}
6621
6622 \keys_define:nn{notesslides / pkg}{
6623   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6624   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6625   notes        .bool_set:N = \c_notesslides_notes_bool ,
6626   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6627   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6628   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
6629   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
6630   nopproblems  .bool_set:N = \c_notesslides_nopproblems_bool,
6631   unknown      .code:n      = {
6632     \PassOptionsToClass{\CurrentOption}{stex}
6633     \PassOptionsToClass{\CurrentOption}{tikzinput}
6634   }
6635 }
6636 \ProcessKeysOptions{ notesslides / pkg }
6637 \newif\ifnotes
6638 \bool_if:NTF \c_notesslides_notes_bool {
6639   \notesttrue
6640 }{
6641   \notesfalse
6642 }
6643

```

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

```

6644 \str_if_empty:NTF \c_notesslides_topsect_str {
6645   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
6646 }{
6647   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
6648 }
6649 </package>

```

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

```

6650 <*cls>
6651 \bool_if:NTF \c_notesslides_notes_bool {
6652   \LoadClass{document-structure}
6653 }{
6654   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6655   \newcounter{Item}
6656   \newcounter{paragraph}
6657   \newcounter{subparagraph}
6658   \newcounter{Hfootnote}
6659   \RequirePackage{document-structure}
6660 }

```

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

```

6661 \RequirePackage{notesslides}
6662 </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.

```

6663 <*package>
6664 \bool_if:NT \c__notesslides_notes_bool {
6665   \RequirePackage{a4wide}
6666   \RequirePackage{marginnote}
6667   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6668   \RequirePackage{mdframed}
6669   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6670   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6671 }
6672 \RequirePackage{stex-tikzinput}
6673 \RequirePackage{etoolbox}
6674 \RequirePackage{amssymb}
6675 \RequirePackage{amsmath}
6676 \RequirePackage{comment}
6677 \RequirePackage{textcomp}
6678 \RequirePackage{url}
6679 \RequirePackage{graphicx}
6680 \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

```

6681 \bool_if:NT \c__notesslides_notes_bool {
6682   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6683 }
6684
6685
6686 \NewDocumentCommand \libusetheme {0{} m} {
6687   \bool_if:NTF \c__notesslides_notes_bool {
6688     \libusepackage[#1]{beamernotestheme#2}
6689   }{
6690     \libusepackage[#1]{beamertheme#2}
6691   }
6692 }

```

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

```

6693 \newcounter{slide}
6694 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6695 \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.

```

6696 \bool_if:NTF \c__notesslides_notes_bool {
6697   \renewenvironment{note}{\ignorespaces}{}
6698 }{
6699   \excludecomment{note}
6700 }

```

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.

```

6701 \bool_if:NT \c__notesslides_notes_bool {
6702   \newlength{\slideframewidth}
6703   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6704 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6705   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6706     \bool_set_true:N #1
6707   }{
6708     \bool_set_false:N #1
6709   }
6710 }
6711 \keys_define:nn{notesslides / frame}{
6712   label .str_set_x:N = \l__notesslides_frame_label_str,
6713   allowframebreaks .code:n = {
6714     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6715   },
6716   allowdisplaybreaks .code:n = {
6717     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6718   },
6719   fragile .code:n = {
6720     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6721   },
6722   shrink .code:n = {
6723     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6724   },
6725   squeeze .code:n = {
6726     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6727   },
6728   t .code:n = {
6729     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6730   },
6731 }
6732 \cs_new_protected:Nn \__notesslides_frame_args:n {
6733   \str_clear:N \l__notesslides_frame_label_str
6734   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6735   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6736   \bool_set_true:N \l__notesslides_frame_fragile_bool
6737   \bool_set_true:N \l__notesslides_frame_shrink_bool
6738   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6739   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6740 \keys_set:nn { notesslides / frame }{ #1 }
6741 }

```

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

```

6742 \renewenvironment{frame}[1][]{
6743   \__notesslides_frame_args:n{#1}
6744   \sffamily
6745   \stepcounter{slide}
6746   \def\@currentlabel{\theslide}
6747   \str_if_empty:NF \l__notesslides_frame_label_str {
6748     \label{\l__notesslides_frame_label_str}
6749   }

```

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

```

6750 \def\itemize@level{outer}
6751 \def\itemize@outer{outer}
6752 \def\itemize@inner{inner}
6753 \renewcommand\newpage{\addtocounter{framenumber}{1}}
6754 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6755 \renewenvironment{itemize}{
6756   \ifx\itemize@level\itemize@outer
6757     \def\itemize@label{$\rhd$}
6758   \fi
6759   \ifx\itemize@level\itemize@inner
6760     \def\itemize@label{$\scriptstyle\rhd$}
6761   \fi
6762   \begin{list}
6763     {\itemize@label}
6764     {\setlength{\labelsep}{.3em}
6765      \setlength{\labelwidth}{.5em}
6766      \setlength{\leftmargin}{1.5em}
6767     }
6768   \edef\itemize@level{\itemize@inner}
6769 }{
6770   \end{list}
6771 }

```

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

```

6772 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6773 }{
6774   \medskip\miko@slidelabel\end{mdframed}
6775 }

```

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

`\frametitle`

```

6776 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6777 }

```

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

EdN:15

`\pause`

```

15
6778 \bool_if:NT \c__notesslides_notes_bool {
6779   \newcommand\pause{}
6780 }

```

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

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

nparagraph

```
6781 \bool_if:NTF \c__notesslides_notes_bool {  
6782   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}  
6783 }{  
6784   \excludecomment{nparagraph}  
6785 }
```

nfragment

```
6786 \bool_if:NTF \c__notesslides_notes_bool {  
6787   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}  
6788 }{  
6789   \excludecomment{nfragment}  
6790 }
```

ndefinition

```
6791 \bool_if:NTF \c__notesslides_notes_bool {  
6792   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}  
6793 }{  
6794   \excludecomment{ndefinition}  
6795 }
```

nassertion

```
6796 \bool_if:NTF \c__notesslides_notes_bool {  
6797   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}  
6798 }{  
6799   \excludecomment{nassertion}  
6800 }
```

nsproof

```
6801 \bool_if:NTF \c__notesslides_notes_bool {  
6802   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}  
6803 }{  
6804   \excludecomment{nsproof}  
6805 }
```

nexample

```
6806 \bool_if:NTF \c__notesslides_notes_bool {  
6807   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}  
6808 }{  
6809   \excludecomment{nexample}  
6810 }
```

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

```
6811 \def\inputref@preskip{\smallskip}  
6812 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6813 \let\orig@inputref\inputref
6814 \def\inputref{\@ifstar\ninputref\orig@inputref}
6815 \newcommand\ninputref[2][]{
6816   \bool_if:NT \c__notesslides_notes_bool {
6817     \orig@inputref[#1]{#2}
6818   }
6819 }
```

(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>}`.

```
6820 \newlength{\slidelogoheight}
6821
6822 \bool_if:NTF \c__notesslides_notes_bool {
6823   \setlength{\slidelogoheight}{.4cm}
6824 }{
6825   \setlength{\slidelogoheight}{1cm}
6826 }
6827 \newsavebox{\slidelogo}
6828 \sbox{\slidelogo}{\text{\TeX}}
6829 \newrobustcmd{\setslidelogo}[1]{\def\slidelogo{#1}}
6830 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6831 }
```

(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.

```
6832 \def\source{Michael Kohlhase}% customize locally
6833 \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.

```
6834 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6835 \newsavebox{\cclogo}
6836 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6837 \newif\ifcchref\cchreffalse
6838 \AtBeginDocument{
6839   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6840 }
6841 \def\licensing{
6842   \ifcchref
```

```

6843 \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6844 \else
6845   {\usebox{\cclogo}}
6846 \fi
6847 }
6848 \newrobustcmd{\setlicensing}[2][]{
6849   \def\@url{\#1}
6850   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{\#2}}
6851   \ifx\@url\@empty
6852     \def\licensing{\usebox{\cclogo}}
6853   \else
6854     \def\licensing{
6855       \ifcchref
6856         \href{\#1}{\usebox{\cclogo}}
6857       \else
6858         {\usebox{\cclogo}}
6859       \fi
6860     }
6861   \fi
6862 }

```

(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

```

6863 \newrobustcmd\miko@slidelabel{
6864   \vbox to \slidelogoheight{
6865     \vss\hbox to \slidewidth
6866     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6867   }
6868 }

```

(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.

```

6869 \def\Gin@mhrepos{}
6870 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{\#1}}
6871 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{\#1}}
6872 \newrobustcmd\frameimage[2][]{
6873   \stepcounter{slide}
6874   \bool_if:NT \c__notesslides_frameimages_bool {
6875     \def\Gin@ewidth{}\setkeys{Gin}{\#1}
6876     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6877     \begin{center}
6878       \bool_if:NTF \c__notesslides_fboxed_bool {
6879         \fbox{
6880           \ifx\Gin@ewidth\@empty
6881             \ifx\Gin@mhrepos\@empty
6882               \mhgraphics[width=\slidewidth,\#1]{\#2}
6883             \else

```

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

```

6884         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6885     \fi
6886 \else% Gin@ewidth empty
6887     \ifx\Gin@mhrepos\@empty
6888         \mhgraphics[#1]{#2}
6889     \else
6890         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6891     \fi
6892 \fi% Gin@ewidth empty
6893 }
6894 }{
6895     \ifx\Gin@ewidth\@empty
6896     \ifx\Gin@mhrepos\@empty
6897         \mhgraphics[width=\slidewidth,#1]{#2}
6898     \else
6899         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6900     \fi
6901     \ifx\Gin@mhrepos\@empty
6902         \mhgraphics[#1]{#2}
6903     \else
6904         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6905     \fi
6906 \fi% Gin@ewidth empty
6907 }
6908 \end{center}
6909 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6910 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6911 }
6912 } % 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.

```

6913 \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

```

6914 \AddToHook{begindocument}{
6915     \definecolor{green}{rgb}{0,.5,0}
6916     \definecolor{purple}{cmyk}{.3,1,0,.17}
6917 }

```

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`.

```

6918 % \def\STpresent#1{\textcolor{blue}{#1}}
6919 \def\defemph#1{\textcolor{magenta}{#1}}
6920 \def\symrefemph#1{\textcolor{cyan}{#1}}
6921 \def\compemph#1{\textcolor{blue}{#1}}
6922 \def\titleemph#1{\textcolor{blue}{#1}}
6923 \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.

```

6924 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6925 \def\smalltextwarning{
6926   \pgfuseimage{miko@small@dbend}
6927   \xspace
6928 }
6929 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6930 \newrobustcmd\textwarning{
6931   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6932   \xspace
6933 }
6934 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6935 \newrobustcmd\bigtextwarning{
6936   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6937   \xspace
6938 }

```

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

```

6939 \newrobustcmd\putgraphicsat[3]{
6940   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6941 }
6942 \newrobustcmd\putat[2]{
6943   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6944 }

```

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`.

```

6945 \bool_if:NT \c__notesslides_sectocframes_bool {
6946   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6947     \newcounter{chapter}\counterwithin*{section}{chapter}
6948   }{
6949     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6950       \newcounter{chapter}\counterwithin*{section}{chapter}
6951     }
6952   }
6953 }

```

`\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
6954 \def\part@prefix{}
6955 \@ifpackageloaded{document-structure}{}{
6956   \str_case:VnF \__notesslidesstopsect {
6957     {part}{
6958       \int_set:Nn \l_document_structure_section_level_int {0}
6959       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6960     \def\part@prefix{\arabic{chapter}.}
6961   }
6962   {chapter}{
6963     \int_set:Nn \l_document_structure_section_level_int {1}
6964     \def\thesection{\arabic{chapter}.\arabic{section}}
6965     \def\part@prefix{\arabic{chapter}.}
6966   }
6967   }{
6968     \int_set:Nn \l_document_structure_section_level_int {2}
6969     \def\part@prefix{}
6970   }
6971 }
6972
6973 \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

```

6974 \renewenvironment{sfragment}[2][]{
6975   \_document_structure_omgroup_args:n { #1 }
6976   \int_incr:N \l_document_structure_section_level_int
6977   \bool_if:NT \c__notesslides_sectocframes_bool {
6978     \stepcounter{slide}
6979     \begin{frame}[noframenumbering]
6980     \vfill\Large\centering
6981     \red{
6982       \ifcase\l_document_structure_section_level_int\or
6983         \stepcounter{part}
6984         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6985         \def\currentsectionlevel{\omdoc@part@kw}
6986       \or
6987         \stepcounter{chapter}
6988         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6989         \def\currentsectionlevel{\omdoc@chapter@kw}
6990       \or
6991         \stepcounter{section}
6992         \def\_notesslideslabel{\part@prefix\arabic{section}}
6993         \def\currentsectionlevel{\omdoc@section@kw}
6994       \or
6995         \stepcounter{subsection}
6996         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6997         \def\currentsectionlevel{\omdoc@subsection@kw}
6998       \or
6999         \stepcounter{subsubsection}
7000         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7001         \def\currentsectionlevel{\omdoc@subsubsection@kw}
7002       \or
7003         \stepcounter{paragraph}
7004         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
7005         \def\currentsectionlevel{\omdoc@paragraph@kw}
7006       \else
7007         \def\_notesslideslabel{}

```



```

7008         \def\currentsectionlevel{\omdoc@paragraph@kw}
7009         \fi% end ifcase
7010         \_notesslideslabel%\sref@label@id\_notesslideslabel
7011         \quad #2%
7012     }%
7013     \vfill%
7014     \end{frame}%
7015 }
7016 \str_if_empty:NF \l__document_structure_omgroup_id_str {
7017     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
7018 }
7019 }{}
7020 }

```

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

```

7021 \def\inserttheorembodyfont{\normalfont}
7022 %\bool_if:NF \c__notesslides_notes_bool {
7023 % \defbeamertemplate{theorem begin}{miko}
7024 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
7025 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
7026 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
7027 % \defbeamertemplate{theorem end}{miko}{\}

```

and we set it as the default one.

```

7028 % \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.

```

7029 % \expandafter\def\csname Parent2\endcsname{}
7030 %}
7031
7032 \AddToHook{begindocument}{% this does not work for some reason
7033     \setbeamertemplate{theorems}[ams style]
7034 }
7035 \bool_if:NT \c__notesslides_notes_bool {
7036     \renewenvironment{columns}[1][\]{}%
7037     \par\noindent%
7038     \begin{minipage}%
7039     \slidewidth\centering\leavevmode%
7040 }{}%
7041     \end{minipage}\par\noindent%
7042 }%
7043 \newsavebox\columnbox%
7044 \renewenvironment<>{column}[2][\]{}%
7045     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
7046 }{}%
7047     \end{minipage}\end{lrbox}\usebox\columnbox%
7048 }%
7049 }
7050 \bool_if:NFT \c__notesslides_noproblems_bool {
7051     \newenvironment{problems}{}{}
7052 }{
7053     \excludecomment{problems}
7054 }

```

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.

```

7055 \gdef\printexcursions{}
7056 \newcommand\excursionref[2]{% label, text
7057   \bool_if:NT \c__notesslides_notes_bool {
7058     \begin{sparagraph}[title=Excursion]
7059       #2 \sref[fallback=the appendix]{#1}.
7060     \end{sparagraph}
7061   }
7062 }
7063 \newcommand\activate@excursion[2][]{
7064   \gappto\printexcursions{\inputref{#1}{#2}}
7065 }
7066 \newcommand\excursion[4][]{% repos, label, path, text
7067   \bool_if:NT \c__notesslides_notes_bool {
7068     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7069   }
7070 }

```

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

`\excursiongroup`

```

7071 \keys_define:nn{notesslides / excursiongroup }{
7072   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7073   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7074   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7075 }
7076 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7077   \tl_clear:N \l__notesslides_excursion_intro_tl
7078   \str_clear:N \l__notesslides_excursion_id_str
7079   \str_clear:N \l__notesslides_excursion_mhrepos_str
7080   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7081 }
7082 \newcommand\excursiongroup[1][]{
7083   \__notesslides_excursion_args:n{ #1 }
7084   \ifdefempty\printexcursions{}% only if there are excursions
7085     {\begin{note}
7086       \begin{sfragment}[#1]{Excursions}%
7087       \ifdefempty\l__notesslides_excursion_intro_tl{\{
7088         \inputref[\l__notesslides_excursion_mhrepos_str]{
7089           \l__notesslides_excursion_intro_tl
7090         }
7091       }
7092       \printexcursions%
7093     \end{sfragment}
7094   \end{note}}
7095 }
7096 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7097 \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.

```
7098 <*package>
7099 <@@=problems>
7100 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7101 \RequirePackage{l3keys2e,stex}
7102
7103 \keys_define:nn { problem / pkg }{
7104   notes      .default:n    = { true },
7105   notes      .bool_set:N   = \c__problems_notes_bool,
7106   gnotes     .default:n    = { true },
7107   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7108   hints      .default:n    = { true },
7109   hints      .bool_set:N   = \c__problems_hints_bool,
7110   solutions  .default:n    = { true },
7111   solutions  .bool_set:N   = \c__problems_solutions_bool,
7112   pts        .default:n    = { true },
7113   pts        .bool_set:N   = \c__problems_pts_bool,
7114   min        .default:n    = { true },
7115   min        .bool_set:N   = \c__problems_min_bool,
7116   boxed      .default:n    = { true },
7117   boxed      .bool_set:N   = \c__problems_boxed_bool,
7118   unknown    .code:n       = {}
7119 }
7120 \newif\ifsolutions
7121
7122 \ProcessKeysOptions{ problem / pkg }
7123 \bool_if:NTF \c__problems_solutions_bool {
7124   \solutionstrue
7125 }{
7126   \solutionsfalse
7127 }
```

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

```
7128 \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.

```
7129 \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.

```
7130 \def\prob@problem@kw{Problem}
7131 \def\prob@solution@kw{Solution}
7132 \def\prob@hint@kw{Hint}
7133 \def\prob@note@kw{Note}
7134 \def\prob@gnote@kw{Grading}
7135 \def\prob@pt@kw{pt}
7136 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7137 \AddToHook{begindocument}{
7138   \ltx@ifpackageloaded{babel}{
7139     \makeatletter
7140     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7141     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7142       \input{problem-ngerman.ldf}
7143     }
7144     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7145       \input{problem-finnish.ldf}
7146     }
7147     \clist_if_in:NnT \l_tmpa_clist {french}{
7148       \input{problem-french.ldf}
7149     }
7150     \clist_if_in:NnT \l_tmpa_clist {russian}{
7151       \input{problem-russian.ldf}
7152     }
7153     \makeatother
7154   }{}
7155 }
```

39.2 Problems and Solutions

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

```
7156 \keys_define:nn{ problem / problem }{
7157   id      .str_set_x:N = \l__problems_prob_id_str,
7158   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7159   min     .tl_set:N    = \l__problems_prob_min_tl,
7160   title   .tl_set:N    = \l__problems_prob_title_tl,
7161   type    .tl_set:N    = \l__problems_prob_type_tl,
7162   refnum  .int_set:N    = \l__problems_prob_refnum_int
7163 }
7164 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

7165 \str_clear:N \l__problems_prob_id_str
7166 \tl_clear:N \l__problems_prob_pts_tl
7167 \tl_clear:N \l__problems_prob_min_tl
7168 \tl_clear:N \l__problems_prob_title_tl
7169 \tl_clear:N \l__problems_prob_type_tl
7170 \int_zero_new:N \l__problems_prob_refnum_int
7171 \keys_set:nn { problem / problem }{ #1 }
7172 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7173   \let\l__problems_prob_refnum_int\undefined
7174 }
7175 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7176 \newcounter{problem}
7177 \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.

```

7178 \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

```

7179 \newcommand\prob@number{
7180   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7181     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7182   }{
7183     \int_if_exist:NTF \l__problems_prob_refnum_int {
7184       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7185     }{
7186       \prob@label\theproblem
7187     }
7188   }
7189 }

```

(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.

```

7190 \newcommand\prob@title[3]{%
7191   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7192     #2 \l__problems_inclprob_title_tl #3
7193   }{
7194     \tl_if_exist:NTF \l__problems_prob_title_tl {
7195       #2 \l__problems_prob_title_tl #3
7196     }{
7197       #1
7198     }
7199   }
7200 }

```

(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.

```

7201 \def\prob@heading{
7202   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7203   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
7204 }

```

(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

```

7205 \newenvironment{sproblem}[1][{}]{
7206   \__problems_prob_args:n{#1}%\sref@target%
7207   \@in@omtexttrue% we are in a statement (for inline definitions)
7208   \stepcounter{problem}\record@problem
7209   \def\current@section@level{\prob@problem@kw}
7210   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7211     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7212   }{
7213     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7214   }
7215   \str_if_exist:NTF \l__problems_inclprob_id_str {
7216     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7217   }{
7218     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7219   }
7220
7221
7222   \clist_set:No \l_tmpa_clist \sproblemtype
7223   \tl_clear:N \l_tmpa_tl
7224   \clist_map_inline:Nn \l_tmpa_clist {
7225     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
7226       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
7227     }
7228   }
7229   \tl_if_empty:NTF \l_tmpa_tl {
7230     \__problems_sproblem_start:
7231   }{
7232     \l_tmpa_tl
7233   }
7234   \stex_ref_new_doc_target:n \sproblemid
7235 }{
7236   \clist_set:No \l_tmpa_clist \sproblemtype
7237   \tl_clear:N \l_tmpa_tl
7238   \clist_map_inline:Nn \l_tmpa_clist {
7239     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
7240       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
7241     }

```

```

7242 }
7243 \tl_if_empty:NTF \l_tmpa_tl {
7244   \__problems_sproblem_end:
7245 }{
7246   \l_tmpa_tl
7247 }
7248
7249
7250 \smallskip
7251 }
7252
7253
7254 \cs_new_protected:Nn \__problems_sproblem_start: {
7255   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7256 }
7257 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7258
7259 \newcommand\stexpatchproblem[3][] {
7260   \str_set:Nx \l_tmpa_str{ #1 }
7261   \str_if_empty:NTF \l_tmpa_str {
7262     \tl_set:Nn \__problems_sproblem_start: { #2 }
7263     \tl_set:Nn \__problems_sproblem_end: { #3 }
7264   }{
7265     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7266     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7267   }
7268 }
7269
7270
7271 \bool_if:NT \c__problems_boxed_bool {
7272   \surroundwithmdframed{problem}
7273 }

```

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

```

7274 \def\record@problem{
7275   \protected@write\@auxout{}
7276   {
7277     \string\@problem{\prob@number}
7278     {
7279       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7280         \l__problems_inclprob_pts_tl
7281       }{
7282         \l__problems_prob_pts_tl
7283       }
7284     }%
7285     {
7286       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7287         \l__problems_inclprob_min_tl
7288       }{
7289         \l__problems_prob_min_tl
7290       }
7291     }
7292   }
7293 }

```

(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).

```
7294 \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.

```
7295 \keys_define:nn { problem / solution }{
7296   id                .str_set_x:N = \l__problems_solution_id_str ,
7297   for               .tl_set:N   = \l__problems_solution_for_tl ,
7298   height            .dim_set:N  = \l__problems_solution_height_dim ,
7299   creators          .clist_set:N = \l__problems_solution_creators_clist ,
7300   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
7301   srccite           .tl_set:N   = \l__problems_solution_srccite_tl
7302 }
7303 \cs_new_protected:Nn \__problems_solution_args:n {
7304   \str_clear:N \l__problems_solution_id_str
7305   \tl_clear:N \l__problems_solution_for_tl
7306   \tl_clear:N \l__problems_solution_srccite_tl
7307   \clist_clear:N \l__problems_solution_creators_clist
7308   \clist_clear:N \l__problems_solution_contributors_clist
7309   \dim_zero:N \l__problems_solution_height_dim
7310   \keys_set:nn { problem / solution }{ #1 }
7311 }
```

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

```
7312 \newcommand\@startsolution[1][{}]{
7313   \__problems_solution_args:n { #1 }
7314   \@in@omtexttrue% we are in a statement.
7315   \bool_if:NF \c__problems_boxed_bool { \hrule }
7316   \smallskip\noindent
7317   {\textbf\prob@solution@kw : \enspace}
7318   \begin{small}
7319   \def\current@section@level{\prob@solution@kw}
7320   \ignorespacesandpars
7321 }
```

\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.

```
7322 \newcommand\startsolutions{
7323   \specialcomment{solution}{\@startsolution}{
7324     \bool_if:NF \c__problems_boxed_bool {
7325       \hrule\medskip
7326     }
7327     \end{small}%
7328   }
7329   \bool_if:NT \c__problems_boxed_bool {
7330     \surroundwithmdframed{solution}
7331   }
7332 }
```


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

\stopsolutions

```
7333 \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.

```
7334 \ifsolutions
7335   \startsolutions
7336 \else
7337   \stopsolutions
7338 \fi
```

exnote

```
7339 \bool_if:NTF \c__problems_notes_bool {
7340   \newenvironment{exnote}[1][]{
7341     \par\smallskip\hrule\smallskip
7342     \noindent\textbf{\prob@note@kw : }\small
7343   }{
7344     \smallskip\hrule
7345   }
7346 }{
7347   \excludecomment{exnote}
7348 }
```

hint

```
7349 \bool_if:NTF \c__problems_notes_bool {
7350   \newenvironment{hint}[1][]{
7351     \par\smallskip\hrule\smallskip
7352     \noindent\textbf{\prob@hint@kw :~ }\small
7353   }{
7354     \smallskip\hrule
7355   }
7356 \newenvironment{exhint}[1][]{
7357   \par\smallskip\hrule\smallskip
7358   \noindent\textbf{\prob@hint@kw :~ }\small
7359 }{
7360   \smallskip\hrule
7361 }
7362 }{
7363   \excludecomment{hint}
7364   \excludecomment{exhint}
7365 }
```

gnote

```
7366 \bool_if:NTF \c__problems_notes_bool {
7367   \newenvironment{gnote}[1][]{
7368     \par\smallskip\hrule\smallskip
7369     \noindent\textbf{\prob@gnote@kw : }\small
7370   }{
7371     \smallskip\hrule
7372   }
7373 }{
7374   \excludecomment{gnote}
7375 }
```

39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7376 \newenvironment{mcb}{
7377   \begin{enumerate}
7378 }{
7379   \end{enumerate}
7380 }
```

we define the keys for the mcc macro

```

7381 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7382   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7383     \bool_set_true:N #1
7384   }{
7385     \bool_set_false:N #1
7386   }
7387 }
7388 \keys_define:nn { problem / mcc }{
7389   id          .str_set_x:N = \l__problems_mcc_id_str ,
7390   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7391   T           .default:n   = { true } ,
7392   T           .bool_set:N  = \l__problems_mcc_t_bool ,
7393   F           .default:n   = { true } ,
7394   F           .bool_set:N  = \l__problems_mcc_f_bool ,
7395   Ttext       .code:n      = {
7396     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7397   } ,
7398   Ftext       .code:n      = {
7399     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7400   }
7401 }
7402 \cs_new_protected:Nn \l__problems_mcc_args:n {
7403   \str_clear:N \l__problems_mcc_id_str
7404   \tl_clear:N \l__problems_mcc_feedback_tl
7405   \bool_set_true:N \l__problems_mcc_t_bool
7406   \bool_set_true:N \l__problems_mcc_f_bool
7407   \bool_set_true:N \l__problems_mcc_Ttext_bool
7408   \bool_set_false:N \l__problems_mcc_Ftext_bool
7409   \keys_set:nn { problem / mcc }{ #1 }
7410 }
```

\mcc

```

7411 \newcommand\mcc[2][]{
7412   \l__problems_mcc_args:n{ #1 }
7413   \item #2
7414   \ifsolutions
7415     \l
7416     \bool_if:NT \l__problems_mcc_t_bool {
7417       % TODO!
7418       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7419     }
7420     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

7421      % TODO!
7422      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7423    }
7424    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7425      !
7426    }{
7427      \l__problems_mcc_feedback_tl
7428    }
7429    \fi
7430  } %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.

```

7431
7432 \keys_define:nn{ problem / inclproblem }{
7433   id      .str_set:N = \l__problems_inclprob_id_str,
7434   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7435   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7436   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7437   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7438   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7439   mhrepos .str_set:N = \l__problems_inclprob_mhrepos_str
7440 }
7441 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7442   \str_clear:N \l__problems_prob_id_str
7443   \tl_clear:N \l__problems_inclprob_pts_tl
7444   \tl_clear:N \l__problems_inclprob_min_tl
7445   \tl_clear:N \l__problems_inclprob_title_tl
7446   \tl_clear:N \l__problems_inclprob_type_tl
7447   \int_zero_new:N \l__problems_inclprob_refnum_int
7448   \str_clear:N \l__problems_inclprob_mhrepos_str
7449   \keys_set:nn { problem / inclproblem }{ #1 }
7450   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7451     \let\l__problems_inclprob_pts_tl\undefined
7452   }
7453   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7454     \let\l__problems_inclprob_min_tl\undefined
7455   }
7456   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7457     \let\l__problems_inclprob_title_tl\undefined
7458   }
7459   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7460     \let\l__problems_inclprob_type_tl\undefined
7461   }
7462   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7463     \let\l__problems_inclprob_refnum_int\undefined
7464   }
7465 }

```

```

7466
7467 \cs_new_protected:Nn \__problems_inclprob_clear: {
7468   \let\l__problems_inclprob_id_str\undefined
7469   \let\l__problems_inclprob_pts_tl\undefined
7470   \let\l__problems_inclprob_min_tl\undefined
7471   \let\l__problems_inclprob_title_tl\undefined
7472   \let\l__problems_inclprob_type_tl\undefined
7473   \let\l__problems_inclprob_refnum_int\undefined
7474   \let\l__problems_inclprob_mhrepos_str\undefined
7475 }
7476 \__problems_inclprob_clear:
7477
7478 \newcommand\includeproblem[2][ ]{
7479   \__problems_inclprob_args:n{ #1 }
7480   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7481     \input{#2}
7482   }{
7483     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7484       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7485     }
7486   }
7487   \__problems_inclprob_clear:
7488 }

```

(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.

```

7489 \AddToHook{enddocument}{
7490   \bool_if:NT \c__problems_pts_bool {
7491     \message{Total:~\arabic{pts}~points}
7492   }
7493   \bool_if:NT \c__problems_min_bool {
7494     \message{Total:~\arabic{min}~minutes}
7495   }
7496 }

```

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

```

7497 \def\pts#1{
7498   \bool_if:NT \c__problems_pts_bool {
7499     \marginpar{#1~\prob@pt@kw}
7500   }
7501 }
7502 \def\min#1{
7503   \bool_if:NT \c__problems_min_bool {
7504     \marginpar{#1~\prob@min@kw}
7505   }
7506 }

```

`\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.

```

7507 \newcounter{pts}
7508 \def\show@pts{
7509   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7510     \bool_if:NT \c__problems_pts_bool {
7511       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7512       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7513     }
7514   }{
7515     \tl_if_exist:NT \l__problems_prob_pts_tl {
7516       \bool_if:NT \c__problems_pts_bool {
7517         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7518         \addtocounter{pts}{\l__problems_prob_pts_tl}
7519       }
7520     }
7521   }
7522 }

```

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

`\show@min`

```

7523 \newcounter{min}
7524 \def\show@min{
7525   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7526     \bool_if:NT \c__problems_min_bool {
7527       \marginpar{\l__problems_inclprob_min_tl\ min}
7528       \addtocounter{min}{\l__problems_inclprob_min_tl}
7529     }
7530   }{
7531     \tl_if_exist:NT \l__problems_prob_min_tl {
7532       \bool_if:NT \c__problems_min_bool {
7533         \marginpar{\l__problems_prob_min_tl\ min}
7534         \addtocounter{min}{\l__problems_prob_min_tl}
7535       }
7536     }
7537   }
7538 }
7539 </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.

```
7540 \@@=hwexam>
7541 \*cls>
7542 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7543 \RequirePackage{l3keys2e}
7544 \DeclareOption*{
7545   \PassOptionsToClass{\CurrentOption}{document-structure}
7546   \PassOptionsToPackage{\CurrentOption}{stex}
7547   \PassOptionsToPackage{\CurrentOption}{hwexam}
7548   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7549 }
7550 \ProcessOptions
```

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

```
7551 \LoadClass{document-structure}
7552 \RequirePackage{stex}
7553 \RequirePackage{hwexam}
7554 \RequirePackage{tikzinput}
7555 \RequirePackage{graphicx}
7556 \RequirePackage{a4wide}
7557 \RequirePackage{amssymb}
7558 \RequirePackage{amstext}
7559 \RequirePackage{amsmath}
```

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

```

7560 \newcommand\assig@default@type{\hwexam@assignment@kw}
7561 \def\document@hwexamtype{\assig@default@type}
7562 <@@=document_structure>
7563 \keys_define:nn { document-structure / document }{
7564 id .str_set_x:N = \c_document_structure_document_id_str,
7565 hwexamtype .tl_set:N = \document@hwexamtype
7566 }
7567 <@@=hwexam>
7568 </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.

```
7569 \*package>
7570 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7571 \RequirePackage{13keys2e}
7572
7573 \newif\iftest\testfalse
7574 \DeclareOption{test}{\testtrue}
7575 \newif\ifmultiple\multiplefalse
7576 \DeclareOption{multiple}{\multipletrue}
7577 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7578 \ProcessOptions
```

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

```
7579 \RequirePackage{keyval}[1997/11/10]
7580 \RequirePackage{problem}
```

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

```
7581 \newcommand\hwexam@assignment@kw{Assignment}
7582 \newcommand\hwexam@given@kw{Given}
7583 \newcommand\hwexam@due@kw{Due}
7584 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7585 blank~for~extra~space}
7586 \def\hwexam@minutes@kw{minutes}
7587 \newcommand\correction@probs@kw{prob.}
7588 \newcommand\correction@pts@kw{total}
7589 \newcommand\correction@reached@kw{reached}
7590 \newcommand\correction@sum@kw{Sum}
7591 \newcommand\correction@grade@kw{grade}
7592 \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

```

7593 \AddToHook{begindocument}{
7594 \ltx@ifpackageloaded{babel}{
7595 \makeatletter
7596 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7597 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7598 \input{hwexam-ngerman.ldf}
7599 }
7600 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7601 \input{hwexam-finnish.ldf}
7602 }
7603 \clist_if_in:NnT \l_tmpa_clist {french}{
7604 \input{hwexam-french.ldf}
7605 }
7606 \clist_if_in:NnT \l_tmpa_clist {russian}{
7607 \input{hwexam-russian.ldf}
7608 }
7609 \makeatother
7610 }{}
7611 }
7612

```

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.

```

7613 \newcounter{assignment}
7614 \numberproblemsin{assignment}
7615 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7616 \keys_define:nn { hwexam / assignment } {
7617 id .str_set:N = \l__hwexam_assign_id_str,
7618 number .int_set:N = \l__hwexam_assign_number_int,
7619 title .tl_set:N = \l__hwexam_assign_title_tl,
7620 type .tl_set:N = \l__hwexam_assign_type_tl,
7621 given .tl_set:N = \l__hwexam_assign_given_tl,
7622 due .tl_set:N = \l__hwexam_assign_due_tl,
7623 loadmodules .code:n = {
7624 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7625 }
7626 }
7627 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7628 \str_clear:N \l__hwexam_assign_id_str
7629 \int_set:Nn \l__hwexam_assign_number_int {-1}
7630 \tl_clear:N \l__hwexam_assign_title_tl
7631 \tl_clear:N \l__hwexam_assign_type_tl
7632 \tl_clear:N \l__hwexam_assign_given_tl
7633 \tl_clear:N \l__hwexam_assign_due_tl
7634 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7635 \keys_set:nn { hwexam / assignment }{ #1 }
7636 }

```

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.

```

7637 \newcommand\given@due[2]{
7638 \bool_lazy_all:nF {
7639 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7640 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7641 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7642 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7643 }{ #1 }
7644
7645 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7646 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7647 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7648 }
7649 }{
7650 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7651 }
7652
7653 \bool_lazy_or:nnF {
7654 \bool_lazy_and_p:nn {
7655 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7656 }{
7657 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7658 }
7659 }{
7660 \bool_lazy_and_p:nn {
7661 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7662 }{
7663 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7664 }
7665 }{ ,~ }
7666
7667 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7668 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7669 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7670 }
7671 }{
7672 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7673 }
7674
7675 \bool_lazy_all:nF {
7676 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7677 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7678 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7679 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7680 }{ #2 }
7681 }

```

`\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.

```

7682 \newcommand\assignment@title[3]{
7683 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7684 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7685 #1
7686 }{
7687 #2\l__hwexam_assign_title_tl#3
7688 }
7689 }{
7690 #2\l__hwexam_inclasssign_title_tl#3
7691 }
7692 }

```

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

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

```

7693 \newcommand\assignment@number{
7694 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7695 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7696 \arabic{assignment}
7697 } {
7698 \int_use:N \l__hwexam_assign_number_int
7699 }
7700 }{
7701 \int_use:N \l__hwexam_inclasssign_number_int
7702 }
7703 }

```

(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.

```

7704 \newenvironment{assignment}[1][ ]{
7705 \__hwexam_assignment_args:n { #1 }
7706 %\sref@target
7707 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7708 \global\stepcounter{assignment}
7709 }{
7710 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7711 }
7712 \setcounter{problem}{0}
7713 \def\current@section@level{\document@hwexamtype}
7714 %\sref@label@id{\document@hwexamtype \thesection}
7715 \begin{@assignment}
7716 }{
7717 \end{@assignment}
7718 }

```

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

```

7719 \def\ass@title{
7720 \protect\document@hwexamtype~\arabic{assignment}
7721 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7722 }
7723 \ifmultiple
7724 \newenvironment{@assignment}{
7725 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7726 \begin{sfragment}[loadmodules]{\ass@title}
7727 }{
7728 \begin{sfragment}{\ass@title}
7729 }
7730 }{
7731 \end{sfragment}
7732 }

```

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

```

7733 \else
7734 \newenvironment{@assignment}{
7735 \begin{center}\bf
7736 \Large@title\strut\
7737 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7738 \large\given@due{--\}{}\{;\}{}
7739 \end{center}
7740 }{}
7741 \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.

```

7742 \keys_define:nn { hwexam / inclassignment } {
7743 %id .str_set_x:N = \l__hwexam_assign_id_str,
7744 number .int_set:N = \l__hwexam_inclassign_number_int,
7745 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7746 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7747 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7748 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7749 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7750 }
7751 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7752 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7753 \tl_clear:N \l__hwexam_inclassign_title_tl
7754 \tl_clear:N \l__hwexam_inclassign_type_tl
7755 \tl_clear:N \l__hwexam_inclassign_given_tl
7756 \tl_clear:N \l__hwexam_inclassign_due_tl
7757 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7758 \keys_set:nn { hwexam / inclassignment }{ #1 }
7759 }
7760 \__hwexam_inclassignment_args:n {}
7761
7762 \newcommand\inputassignment[2][{}]{

```

```

7763 \_hwexam_inclassnment_args:n { #1 }
7764 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7765 \input{#2}
7766 }{
7767 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7768 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7769 }
7770 }
7771 \_hwexam_inclassnment_args:n {}
7772 }
7773 \newcommand\includeassignment[2][]{
7774 \newpage
7775 \inputassignment[#1]{#2}
7776 }

```

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

41.4 Typesetting Exams

\quizheading

```

7777 \ExplSyntaxOff
7778 \newcommand\quizheading[1]{%
7779 \def\@tas{#1}%
7780 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7781 \ifx\@tas\@empty\else%
7782 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7783 \fi%
7784 }
7785 \ExplSyntaxOn

```

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

\testheading

```

7786
7787 \def\hwexamheader{\input{hwexam-default.header}}
7788
7789 \def\hwexamminutes{
7790 \tl_if_empty:NTF \testheading@duration {
7791 {\testheading@min}~\hwexam@minutes@kw
7792 }{
7793 \testheading@duration
7794 }
7795 }
7796
7797 \keys_define:nn { hwexam / testheading } {
7798 min .tl_set:N = \testheading@min,
7799 duration .tl_set:N = \testheading@duration,
7800 reqpts .tl_set:N = \testheading@reqpts,
7801 tools .tl_set:N = \testheading@tools
7802 }
7803 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7804 \tl_clear:N \testheading@min
7805 \tl_clear:N \testheading@duration

```

```

7806 \tl_clear:N \testheading@reqpts
7807 \tl_clear:N \testheading@tools
7808 \keys_set:nn { hwexam / testheading }{ #1 }
7809 }
7810 \newenvironment{testheading}[1][]{
7811 \_hwexam_testheading_args:n{ #1 }
7812 \newcount\check@time\check@time=\testheading@min
7813 \advance\check@time by -\theassignment@totalmin
7814 \newif\if@bonuspoints
7815 \tl_if_empty:NTF \testheading@reqpts {
7816 \@bonuspointsfalse
7817 }{
7818 \newcount\bonus@pts
7819 \bonus@pts=\theassignment@totalpts
7820 \advance\bonus@pts by -\testheading@reqpts
7821 \edef\bonus@pts{\the\bonus@pts}
7822 \@bonuspointstrue
7823 }
7824 \edef\check@time{\the\check@time}
7825
7826 \makeatletter\hwexamheader\makeatother
7827 }{
7828 \newpage
7829 }

```

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

\testspace

```

7830 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7831 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7832 \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.

```

7833 <@=problems>
7834 \renewcommand\@problem[3]{
7835 \stepcounter{assignment@probs}
7836 \def\__problemspts{#2}
7837 \ifx\__problemspts\@empty\else
7838 \addtocounter{assignment@totalpts}{#2}
7839 \fi
7840 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7841 \xdef\correction@probs{\correction@probs & #1}%
7842 \xdef\correction@pts{\correction@pts & #2}
7843 \xdef\correction@reached{\correction@reached &}

```

```

7844 }
7845 <@@=hwexam>

```

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

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

```

7846 \newcounter{assignment@probs}
7847 \newcounter{assignment@totalpts}
7848 \newcounter{assignment@totalmin}
7849 \def\correction@probs{\correction@probs@kw}
7850 \def\correction@pts{\correction@pts@kw}
7851 \def\correction@reached{\correction@reached@kw}
7852 \stepcounter{assignment@probs}
7853 \newcommand\correction@table{
7854 \resizebox{\textwidth}{!}{%
7855 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7856 &\multicolumn{\theassignment@probs}{c|}|%|
7857 {\footnotesize\correction@forgrading@kw} &\\ \hline
7858 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7859 \correction@pts & \theassignment@totalpts & \\ \hline
7860 \correction@reached & & \[.7cm]\hline
7861 \end{tabular}}
7862 </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}

```