

The sTeX3 Package *

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

2022-03-14

Abstract

sTeX is a collection of L^AT_EX packages 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 2022-03-14)

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	158
30	STEX-Terms Implementation	165
30.1	Symbol Invocations	165
30.2	Terms	172
30.3	Notation Components	176
30.4	Variables	179
30.5	Sequences	180
31	STEX-Structural Features Implementation	182
31.1	Imports with modification	183
31.2	The feature environment	190
31.3	Structure	191
32	STEX-Statements Implementation	201
32.1	Definitions	201
32.2	Assertions	206
32.3	Examples	210
32.4	Logical Paragraphs	212
33	The Implementation	218
33.1	Package Options	218
33.2	Proofs	218
33.3	Justifications	229
34	STEX-Others Implementation	231
35	STEX-Metatheory Implementation	232
36	Tikzinput Implementation	235
37	document-structure.sty Implementation	237
37.1	The document-structure Class	237
37.2	Class Options	237
37.3	Beefing up the document environment	238
37.4	Implementation: document-structure Package	238
37.5	Package Options	238
37.6	Document Structure	240
37.7	Front and Backmatter	243
37.8	Global Variables	245

38 NotesSlides – Implementation	246
38.1 Class and Package Options	246
38.2 Notes and Slides	248
38.3 Header and Footer Lines	252
38.4 Frame Images	253
38.5 Colors and Highlighting	254
38.6 Sectioning	255
38.7 Excursions	258
39 The Implementation	259
39.1 Package Options	259
39.2 Problems and Solutions	260
39.3 Multiple Choice Blocks	266
39.4 Including Problems	267
39.5 Reporting Metadata	268
40 Implementation: The hwexam Class	270
40.1 Class Options	270
41 Implementation: The hwexam Package	272
41.1 Package Options	272
41.2 Assignments	273
41.3 Including Assignments	276
41.4 Typesetting Exams	277
41.5 Leftovers	279

Part I

Manual



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



Boxes like this one explain how some \LaTeX concept relates to the MMT/OMDoc system, philosophy or language.

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, $\text{\texttt{\textit{STeX}}}$ expressions are converted to OMDOC/MMT terms, which use the syntax of OPENMATH.

3.2 $\text{\texttt{\textit{STeX}}}$ Archives

3.2.1 The Local MathHub-Directory

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

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

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

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

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

Each such archive needs two subdirectories:

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

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

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

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

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

3.2.3 MANIFEST.MF-Files

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

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

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

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

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

\hookrightarrow An \TeX module corresponds to an MMT/OMDOC *theory*. As such it
 \hookrightarrow gets assigned a module URI (*universal resource identifier*) of the form
 \hookrightarrow `<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 `\smoduletitle`, `\smoduletype` and `\smoduleid`.

For example:

Example 2

Input:

```

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

```

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

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

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

Example 3

Input:

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

Output:

Given a `foo`, we can...

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

Example 4

Input:

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

Output:

`this` is a symbol taking two arguments.

`\notation`

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




Example 5

Input:

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

Output:

First: a ; Second: b

-  `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`

3.3.3 Argument Types

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

b-Type Arguments

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

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

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

Example 9

Input:

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

Output:

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

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

a-Type Arguments

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

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

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

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

Example 10

Input:

```

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

```

Output:

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

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

Example 11

Input:

```

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

```

Output:

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

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

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

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

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

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

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

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

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

B-Type Arguments

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

Example 12

Input:

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

Output:

$\forall x,y,z.P$

3.3.4 Type and Definiens Components

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

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

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

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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



More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

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

In the example above:



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

3.3.6 Variables

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

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

`\svar`

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

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

\vardef

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

Example 19

Input:

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

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

\varseq

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

This is best shown by example:

Example 20

Input:

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

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$$a_1 + \dots + a_n$$

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

Example 22

Input:

```
1 \vardef{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 `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

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

`\STEXexport`

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



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

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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A `monoid` is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

\instantiate

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

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

`\instantiate` and `mathstructure` make use of the *Theories-as-Types* paradigm: `mathstructure{<name>}` does in fact simply create a nested theory with name `<name>-structure`. The *constant* `<name>` is defined as `Mod(<name>-structure)` – a *dependent record type with manifest fields*, the fields of which are generated from (and correspond to) the constants in `<name>-structure`.
 $\hookrightarrow M$ $\hookrightarrow M$ $\rightsquigarrow T$ `\instantiate` appropriately generates a constant whose definiens is a record term of type `Mod(<name>-structure)`, with the fields assigned appropriately based on the key-value-list.

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

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

\varinstantiate

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

This allows us to do things like:

Example 27

Input:

```

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

```

Output:

Test: $a \cdot c$

TODO: explain donotclone

3.4.5 The interpretmodule Environment

TODO: explain

Example 30

Input:

```

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

```

Output:

3.5 Primitive Symbols (The \TeX Metatheory)

TODO: metatheory documentation

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

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

Example 31

Input:

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

Output:

A natural number is...

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

`\Symname`

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

Example 32

Input:

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

Output:

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



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

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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 33

Input:

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

Output:

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

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

\hookrightarrow M \rightarrow As expected, the above example is translated to OMDOC/MMT as an
 \rightarrow M \rightarrow OMA with `<OMS name="...?addition"/>` as head and `<OMV name="n"/>` and
 \rightarrow T \rightarrow `<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 $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>
--

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

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

<code>\stex_set_current_repository:n</code>

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

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

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

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

9.1.3 Using Content in Archives

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

Chapter 10

STEX-References

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

10.1 Macros and Environments

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

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

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

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

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

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

10.1.2 Using References

<u><code>\sref</code></u>	<code>\sref[<i><opt-args></i>]{<i><id></i>}</code>
---------------------------	--

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

<u><code>\srefsym</code></u>	<code>\srefsym[<i><opt-args></i>]{<i><symbol></i>}</code>
------------------------------	---

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

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

<u><code>\srefsymuri</code></u>	<code>\srefsymuri{<i><URI></i>}{<i><text></i>}</code>
---------------------------------	---

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

Chapter 11

sTeX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

 $\backslash\text{c_stex_module_}\langle URI \rangle_prop$

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.

 $\backslash\text{c_stex_module_}\langle URI \rangle_code$

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

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

The names of all constants declared in the module

 $\backslash\text{c_stex_module_}\langle URI \rangle_constants$

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

`\stex_import_module_uri:nn`

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

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

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

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

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

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

17.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

17.2 The User Interface

17.2.1 Package Options

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

17.2.2 Proofs and Proof steps

`sproof` The `proof` environment is the main container for proofs. It takes an optional `KeyVal` argument that allows to specify the `id` (identifier) and `for` (for which assertion is this a proof) keys. The regular argument of the `proof` environment contains an introductory comment, that may be used to announce the proof style. The `proof` environment contains a sequence of `\step`, `proofcomment`, and `pfcases` environments that are used to markup the proof steps. The `proof` environment has a variant `Proof`, which does not use the proof end marker. This is convenient, if a proof ends in a case distinction, which brings it's own proof end marker with it. The `Proof` environment is a variant of `proof` that does not mark the end of a proof with a little box; presumably, since one of the subproofs already has one and then a box supplied by the outer proof would generate an otherwise empty line. The `\spfidea` macro allows to give a one-paragraph description of the proof idea.

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

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

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

17.2.3 Justifications

`justification` This evidence is marked up with the `justification` environment in the `sproof` package. This environment totally invisible to the formatted result; it wraps the text in the proof step that corresponds to the evidence. The environment takes an optional `KeyVal` argument, which can have the `method` key, whose value is the name of a proof method (this will only need to mean something to the application that consumes the semantic annotations). Furthermore, the justification can contain “premises” (specifications to assertions that were used justify the step) and “arguments” (other information taken into account by the proof method).

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

`\justarg` The `\justarg` macro is very similar to `\premise` with the difference that it is used to mark up arguments to the proof method. Therefore the content of the first argument is interpreted as a mathematical object rather than as an identifier as in the case of `\premise`. In our example, we specified that the simplification should take place on the right hand side of the equation. Other examples include proof methods that instantiate. Here we would indicate the substituted object in a `\justarg` macro.

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

1. For the induction we have to consider the following cases:
 - 1.1. $n = 1$: then we compute $1 = 1^2$ □
 - 1.2. $n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$ □
 - 1.3. $n > 1$:
 - 1.3.1. Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.
 - 1.3.2. We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.
 - 1.3.3. We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum
 - 1.3.4. Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.
 - 1.3.5. We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion. □
 - 1.4. We have considered all the cases, so we have proven the assertion. □

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

17.2.4 Proof Structure

subproof	The <code>pfcases</code> environment is used to mark up a subproof. This environment takes an optional <code>KeyVal</code> argument for semantic annotations and a second argument that allows
method	to specify an introductory comment (just like in the <code>proof</code> environment). The <code>method</code> key can be used to give the name of the proof method executed to make this subproof.
spfcases	The <code>pfcases</code> environment is used to mark up a proof by cases. Technically it is a variant of the <code>subproof</code> where the <code>method</code> is <code>by-cases</code> . Its contents are <code>spfcase</code> environments that mark up the cases one by one.
spfcase	The content of a <code>pfcases</code> environment are a sequence of case proofs marked up in the <code>pfcase</code> environment, which takes an optional <code>KeyVal</code> argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a <code>pfcase</code> environment is the same as that of a <code>proof</code> , i.e.
\spfcasesketch	<code>steps</code> , <code>proofcomments</code> , and <code>pfcases</code> environments. <code>\spfcasesketch</code> is a variant of the <code>spfcase</code> environment that takes the same arguments, but instead of the <code>spfsteps</code> in the body uses a third argument for a proof sketch.
sproofcomment	The <code>sproofcomment</code> environment is much like a <code>step</code> , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a <code>\premise</code> .

17.2.5 Proof End Markers

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

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

17.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

EdN:3

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 T E X collection, a version of T E X/L A T E X that allows to markup T E X/L A T E X documents semantically without leaving the document format, essentially turning T E X/L A T E X into a document format for mathematical knowledge management (MKM).

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

20.1 Introduction

\S T E X is a version of T E X/L A T E X that allows to markup T E X/L A T E X documents semantically without leaving the document format, essentially turning T E X/L A T E X 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 T E X sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the \S T E X collection.

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

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-strcture` class accept the following options:

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

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

20.2.2 Document Structure

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

```
\begin{smodule}{foo}
\symdef{bar}{B^a_r}
...
\begin{sfragment}[id=sec.barderiv,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[\langle URL \rangle]{\langle label \rangle}`, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL `\langle URL \rangle` that lets LATEXML generate the correct reference.

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

`\STRsemantics`

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{\langle vname \rangle}{\langle text \rangle}` to set the global variable `\langle vname \rangle` to `\langle text \rangle` and `\useSGvar{\langle vname \rangle}` to reference it.

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

`\useSGvar`

`\ifSGvar`

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 `noframenumbers` option so that the frame numbering is kept in sync between the slides and the course notes.

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

If we want to transclude a the contents of a file as a note, we can use a new variant `\inputref*` of the `\inputref` macro from [KGA20]: `\inputref*{foo}` is equivalent to `\begin{note}\inputref{foo}\end{note}`.

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

`\inputref*`
`nparagraph`
`nfragment`
`ndefinition`
`nexample`
`nsproof`
`nassertion`

21.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `notesslides` package mentions copyright and licensing. In the `beamer` class, `\source` stores the author's name as the copyright holder. By default it is *Michael Kohlhase* in the `notesslides` package since he is the main user and designer of this package. `\setsource{<name>}` can change the writer's name. For licensing, we use the Creative Commons Attribution-ShareAlike license by default to strengthen the public domain. If package `hyperref` is loaded, then we can attach a hyperlink to the license logo. `\setlicensing[<url>]{<logo name>}` is used for customization, where `<url>` is optional.

`\setslidelogo`
`\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.

`number`
`title`
`type`
`given`
`due`

23.3 Limitations

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

1. none reported yet.

Part IV

Implementation

Chapter 24

ST_EX -Basics Implementation

24.1 The ST_EXDocument Class

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

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

24.2 Preliminaries

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

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

```

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

```

24.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for L^AT_EXML:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`

`\latexml_if:TF`

```

80 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
81   \if@latexml
82     \prg_return_true:
83   \else:
84     \prg_return_false:
85   \fi:
86 }

```

(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.

```

87 \tl_new:N \l__stex_annotate_arg_tl
88 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
89   \rustex_if:TF {
90     \rustex_direct_HTML:n { \c_ampersand_str \c_hash_str 8205; }
91   }{~}
92 }

```

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

`_stex_annotate_checkempty:n`

```

93 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
94   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
95   \tl_if_empty:NT \l__stex_annotate_arg_tl {
96     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
97   }
98 }

```


(End definition for `_stex_annotate_checkempty:n`.)

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

```

99 \bool_new:N \_stex_html_do_output_bool
100 \bool_set_true:N \_stex_html_do_output_bool
101
102 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
103   \bool_if:nTF \_stex_html_do_output_bool
104     \prg_return_true: \prg_return_false:
105 }
```

(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

```

106 \cs_new_protected:Nn \stex_suppress_html:n {
107   \exp_args:Nne \use:nn {
108     \bool_set_false:N \_stex_html_do_output_bool
109     #1
110   }{
111     \stex_if_do_html:T {
112       \bool_set_true:N \_stex_html_do_output_bool
113     }
114   }
115 }
```

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

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

`\stex_annotate_invisible:n` The p_DF_LA_TE_X-macros largely do nothing; the R_US_E_T_EX-implementations are pretty clear in what they do, the L^AT_EX_ML-implementations resort to perl bindings.

`\stex_annotate_invisible:nnn`

```

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

```

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

```

```

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

```

(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

```

211 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

212 \prop_const_from_keyval:Nn \c_stex_languages_prop {
213   en = english ,
214   de = ngerman ,
215   ar = arabic ,
216   bg = bulgarian ,
217   ru = russian ,
218   fi = finnish ,
219   ro = romanian ,
220   tr = turkish ,
221   fr = french
222 }
223
224 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
225   english = en ,
226   ngerman = de ,
227   arabic = ar ,
228   bulgarian = bg ,
229   russian = ru ,
230   finnish = fi ,
231   romanian = ro ,
232   turkish = tr ,
233   french = fr
234 }
235 % todo: chinese simplified (zhs)
236 %       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:

```

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

```

24.6 Auxiliary Methods

`\stex_deactivate_macro:Nn`

```

267 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
268   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
269   \def#1{
270     \msg_error:nnnn{stex}{error/deactivated-macro}{\detokenize{#1}}{#2}
271   }
272 }

```

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

`\stex_reactivate_macro:N`

```

273 \cs_new_protected:Nn \stex_reactivate_macro:N {
274   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
275 }

```

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

`\ignorespacesandpars`

```
276 \protected\def\ignorespacesandpars{
277   \begingroup\catcode13=10\relax
278   \@ifnextchar\par{
279     \endgroup\expandafter\ignorespacesandpars\@gobble
280   }{
281     \endgroup
282   }
283 }
```

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

`\MMTrule`

```
284 \NewDocumentCommand \MMTrule {m m}{
285   \seq_set_split:Nnn \l_tmpa_seq , {#2}
286   \int_zero:N \l_tmpa_int
287   \stex_annotate_invisible:nnn{mmtrule}{scala://#1}{
288     $\seq_map_inline:Nn \l_tmpa_seq {
289       \int_incr:N \l_tmpa_int
290       \stex_annotate:nnn{arg}{i\int_use:N \l_tmpa_int}{##1}
291     }$
292   }
293 }
294
295 \NewDocumentCommand \MMTinclude {m}{
296   \stex_annotate_invisible:nnn{import}{#1}{-}
297 }
298 \endpackage
```

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

Chapter 25

STEX -MathHub Implementation

```
299 <*package>
300
301 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
302
303 <@@=stex_path>
304
305 Warnings and error messages
306 \msg_new:nnn{stex}{error/norepository}{
307   No~archive~#1~found~in~#2
308 }
309 \msg_new:nnn{stex}{error/notinarchive}{
310   Not~currently~in~an~archive,~but~\detokenize{#1}~
311   needs~one!
312 }
313 \msg_new:nnn{stex}{error/nofile}{
314   \detokenize{#1}~could~not~find~file~#2
315 }
316 \msg_new:nnn{stex}{error/twofiles}{
317   \detokenize{#1}~found~two~candidates~for~#2
318 }
```

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`

```
317 \cs_new_protected:Nn \stex_path_from_string:Nn {
318   \str_set:Nx \l_tmpa_str { #2 }
319   \str_if_empty:NTF \l_tmpa_str {
320     \seq_clear:N #1
321   }{
322     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
323     \sys_if_platform_windows:T{
324       \seq_clear:N \l_tmpa_tl
```

```

325     \seq_map_inline:Nn #1 {
326       \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
327       \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
328     }
329     \seq_set_eq:NN #1 \l_tmpa_tl
330   }
331   \stex_path_canonicalize:N #1
332 }
333 }
334

```

(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`

```

335 \cs_new_protected:Nn \stex_path_to_string:NN {
336   \exp_args:Nne \str_set:Nn #2 { \seq_use:Nn #1 / }
337 }
338
339 \cs_new:Nn \stex_path_to_string:N {
340   \seq_use:Nn #1 /
341 }

```

(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`

```

342 \str_const:Nn \c__stex_path_dot_str {.}
343 \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.

```

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

```

```

366         \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
367     }
368 }
369 }{
370     \str_if_empty:NF \l_tmpa_tl {
371         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
372     }
373 }
374 }
375 }
376 \seq_gset_eq:NN #1 \l_tmpa_seq
377 }
378 }

```

(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}`

```

379 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
380     \seq_if_empty:NTF #1 {
381         \prg_return_false:
382     }{
383         \seq_get_left:NN #1 \l_tmpa_tl
384         \sys_if_platform_windows:TF{
385             \str_if_in:NnTF \l_tmpa_tl {:}{
386                 \prg_return_true:
387             }{
388                 \prg_return_false:
389             }
390         }{
391             \str_if_empty:NTF \l_tmpa_tl {
392                 \prg_return_true:
393             }{
394                 \prg_return_false:
395             }
396         }
397     }
398 }

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

399 \str_new:N\l_stex_kpsewhich_return_str
400 \cs_new_protected:Nn \stex_kpsewhich:n {
401     \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
402     \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
403     \tl_trim_spaces:N \l_stex_kpsewhich_return_str
404 }

```

(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`

```

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

```

417 <@@=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

```

418 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

419 \str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}
420 \stex_path_from_string:Nn \c_stex_mainfile_seq
421   \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`

```

422 \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`

```

423 \cs_new_protected:Nn \stex_filestack_push:n {
424   \stex_path_from_string:Nn\g_stex_currentfile_seq{#1}
425   \stex_path_if_absolute:NF\g_stex_currentfile_seq{
426     \stex_path_from_string:Nn\g_stex_currentfile_seq{
427       \c_stex_pwd_str/#1
428     }
429   }
430   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
431   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
432 }

```



```

474 }
475 \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
476 \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
477 }

```

(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

```

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

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

494 \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`:

```

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

```

```

513         \bool_set_false:N\l_tmpa_bool
514     }{
515         \file_if_exist:nTF{
516             \stex_path_to_string:N\l_tmpa_seq/meta-inf/MANIFEST.MF
517         }{
518             \seq_put_right:Nn\l_tmpa_seq{meta-inf}
519             \seq_put_right:Nn\l_tmpa_seq{MANIFEST.MF}
520             \bool_set_false:N\l_tmpa_bool
521         }{
522             \seq_pop_right:NN\l_tmpa_seq\l_tmpa_tl
523         }
524     }
525 }
526 }
527 }
528 \seq_set_eq:NN\l__stex_mathhub_manifest_file_seq\l_tmpa_seq
529 }

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

530 \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:

```

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

```

```

559     {ns} {
560       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
561       { ns } \l_tmpb_tl
562     }
563     {dependencies} {
564       \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
565       { deps } \l_tmpb_tl
566     }
567   }{}{}
568 }{}
569 }
570 \ior_close:N \c__stex_mathhub_manifest_ior
571 }

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

572 \cs_new_protected:Nn \stex_set_current_repository:n {
573   \stex_require_repository:n { #1 }
574   \prop_set_eq:Nc \l_stex_current_repository_prop {
575     c_stex_mathhub_#1_manifest_prop
576   }
577 }

```

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

`\stex_require_repository:n`

```

578 \cs_new_protected:Nn \stex_require_repository:n {
579   \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
580     \stex_debug:nn{mathhub}{Opening~archive:~#1}
581     \_stex_mathhub_do_manifest:n { #1 }
582   }
583 }

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

584 %\prop_new:N \l_stex_current_repository_prop
585
586 \_stex_mathhub_find_manifest:N \c_stex_pwd_seq
587 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
588   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
589 } {
590   \_stex_mathhub_parse_manifest:n { main }
591   \prop_get:NnN \c_stex_mathhub_main_manifest_prop {id}
592   \l_tmpa_str
593   \prop_set_eq:cN { c_stex_mathhub\_l_tmpa_str_manifest_prop }
594   \c_stex_mathhub_main_manifest_prop
595   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
596   \stex_debug:nn{mathhub}{Current~repository:~
597     \prop_item:Nn \l_stex_current_repository_prop {id}
598   }
599 }

```

(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.

```

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

```

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

25.5 Using Content in Archives

`\mhpath`

```

638 \def \mhpath #1 #2 {
639   \exp_args:Ne \tl_if_empty:nTF{#1}{
640     \c_stex_mathhub_str /
641     \prop_item:Nn \l_stex_current_repository_prop { id }
642     / source / #2
643   }{
644     \c_stex_mathhub_str / #1 / source / #2

```

```

645 }
646 }

```

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

`\inputref`
`\mhinput`

```

647 \newif \ifinputref \inputreffalse
648
649 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
650   \stex_in_repository:nn {#1} {
651     \ifinputref
652       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
653     \else
654       \inputreftrue
655       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
656     \inputreffalse
657   \fi
658 }
659 }
660 \NewDocumentCommand \mhinput { 0{} m}{
661   \stex_mhinput:nn{ #1 }{ #2 }
662 }
663
664 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
665   \stex_in_repository:nn {#1} {
666     \bool_lazy_any:nTF {
667       {\rustex_if_p:}
668       {\latexml_if_p:}
669     } {
670       \str_clear:N \l_tmpa_str
671       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
672         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
673       }
674       \stex_annotate_invisible:nnn{inputref}{
675         \l_tmpa_str / #2
676       }{}
677     }{
678       \begingroup
679         \inputreftrue
680         \tl_if_empty:nTF{ ##1 }{
681           \input{#2}
682         }{
683           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
684         }
685       \endgroup
686     }
687   }
688 }
689 \NewDocumentCommand \inputref { 0{} m}{
690   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
691 }

```

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

\addmhbibresource

```
692 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {
693   \stex_in_repository:nn {#1} {
694     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
695   }
696 }
697 \newcommand\addmhbibresource[2][]{
698   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }
699 }
```

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

\libinput

```
700 \cs_new_protected:Npn \libinput #1 {
701   \prop_if_exist:NF \l_stex_current_repository_prop {
702     \msg_error:nnn{stex}{error/notinarchive}\libinput
703   }
704   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
705     \msg_error:nnn{stex}{error/notinarchive}\libinput
706   }
707   \seq_clear:N \l__stex_mathhub_libinput_files_seq
708   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
709   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
710
711   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
712     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}
713     \IfFileExists{ \l_tmpa_str }{
714       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
715     }{}
716     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
717     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
718   }
719
720   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}
721   \IfFileExists{ \l_tmpa_str }{
722     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
723   }{}
724
725   \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {
726     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
727   }{
728     \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
729       \input{ ##1 }
730     }
731   }
732 }
```

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

\libusepackage

```
733 \NewDocumentCommand \libusepackage {0{} m} {
734   \prop_if_exist:NF \l_stex_current_repository_prop {
735     \msg_error:nnn{stex}{error/notinarchive}\libusepackage
736   }
```



```

737 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
738   \msg_error:nnn{stex}{error/notinarchive}\libusepackage
739 }
740 \seq_clear:N \l__stex_mathhub_libinput_files_seq
741 \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
742 \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
743
744 \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{
745   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #2}
746   \IfFileExists{ \l_tmpa_str.sty }{
747     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
748   }{
749     \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str
750     \seq_put_right:No \l_tmpa_seq \l_tmpa_str
751   }
752
753   \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #2}
754   \IfFileExists{ \l_tmpa_str.sty }{
755     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
756   }{
757
758     \seq_if_empty:NNTF \l__stex_mathhub_libinput_files_seq {
759       \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
760     }{
761       \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
762         \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
763           \usepackage[#1]{ ##1 }
764         }
765       }{
766         \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
767       }
768     }
769   }

```

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

`\mhgraphics`
`\cmhgraphics`

```

770
771 \AddToHook{begindocument}{
772   \ltx@ifpackageloaded{graphicx}{
773     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
774     \newcommand\mhgraphics[2][{}]{%
775       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
776       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
777       \newcommand\cmhgraphics[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
778     }{

```

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

`\lstinputmhlisting`
`\cmlstinputmhlisting`

```

779 \ltx@ifpackageloaded{listings}{
780   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
781   \newcommand\lstinputmhlisting[2][{}]{%
782     \def\lst@mhrepos{}\setkeys{lst}{#1}%
783     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}

```

```

784     \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
785   }{}
786 }
787
788 \end{package}

```

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

Chapter 26

STEX -References Implementation

```
789 <*package>
790
791 %%%%%%%%%% references.dtx %%%%%%%%%%
792
793 <@@=stex_refs>
794
795 Warnings and error messages
```

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

```
795 %\iow_new:N \c__stex_refs_refs_iow
796 \AddToHook{begindocument}{
797 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
798 }
799 \AddToHook{enddocument}{
800 % \iow_close:N \c__stex_refs_refs_iow
801 }
```

`\STEXreftitle`

```
802 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
803
804 \NewDocumentCommand \STEXreftitle { m } {
805 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
806 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
807 \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:`

```
808 \cs_new_protected:Nn \stex_get_document_uri: {
809   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
810   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
811   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
812   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
813   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
814
815   \str_clear:N \l_tmpa_str
816   \prop_if_exist:NT \l_stex_current_repository_prop {
817     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
818       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
819     }
820   }
821
822   \str_if_empty:NTF \l_tmpa_str {
823     \str_set:Nx \l_stex_current_docns_str {
824       file:/\stex_path_to_string:N \l_tmpa_seq
825     }
826   }{
827     \bool_set_true:N \l_tmpa_bool
828     \bool_while_do:Nn \l_tmpa_bool {
829       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
830       \exp_args:No \str_case:nnTF { \l_tmpb_str } {
831         {source} { \bool_set_false:N \l_tmpa_bool }
832       }{}{
833         \seq_if_empty:NT \l_tmpa_seq {
834           \bool_set_false:N \l_tmpa_bool
835         }
836       }
837     }
838
839     \seq_if_empty:NTF \l_tmpa_seq {
840       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
841     }{
842       \str_set:Nx \l_stex_current_docns_str {
843         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
844       }
845     }
846   }
847 }
```

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

`\l_stex_current_docurl_str`

```
848 \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:`

```
849 \cs_new_protected:Nn \stex_get_document_url: {
850   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
851   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
852   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

853 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
854 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
855
856 \str_clear:N \l_tmpa_str
857 \prop_if_exist:NT \l_stex_current_repository_prop {
858   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
859     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
860       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
861     }
862   }
863 }
864
865 \str_if_empty:NTF \l_tmpa_str {
866   \str_set:Nx \l_stex_current_docurl_str {
867     file:/\stex_path_to_string:N \l_tmpa_seq
868   }
869 }{
870   \bool_set_true:N \l_tmpa_bool
871   \bool_while_do:Nn \l_tmpa_bool {
872     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
873     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
874       {source} { \bool_set_false:N \l_tmpa_bool }
875     }{}{
876       \seq_if_empty:NT \l_tmpa_seq {
877         \bool_set_false:N \l_tmpa_bool
878       }
879     }
880   }
881
882   \seq_if_empty:NTF \l_tmpa_seq {
883     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
884   }{
885     \str_set:Nx \l_stex_current_docurl_str {
886       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
887     }
888   }
889 }
890 }

```

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

26.2 Setting Reference Targets

```

891 \str_const:Nn \c__stex_refs_url_str{URL}
892 \str_const:Nn \c__stex_refs_ref_str{REF}
893 \str_new:N \l__stex_refs_curr_label_str
894 % @currentlabel -> number
895 % @currentlabelname -> title
896 % @currentHref -> name.number <- id of some kind
897 % \theH# -> \arabic{section}
898 % \the# -> number
899 % \hyper@makecurrent{#}
900 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

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

```

(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.

```

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

```

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

```

939 \AtEndDocument{
940   \def\stexauxadddocref#1 #2 {}{}
941 }

```

`\stex_ref_new_sym_target:n`

```

942 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
943   \stex_if_smsmode:TF {
944     \str_if_exist:cF{sref_sym_#1_type}{
945       \stex_get_document_url:
946       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

947     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
948   }
949   }{
950     \str_if_empty:NF \l__stex_refs_curr_label_str {
951       \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
952       \immediate\write\@auxout{
953         \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
954           \l__stex_refs_curr_label_str
955         }
956       }
957     }
958   }
959 }

```

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

26.3 Using References

```

960 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

961
962 \keys_define:nn { stex / sref } {
963   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
964   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
965   pre           .tl_set:N = \l__stex_refs_pre_tl ,
966   post          .tl_set:N = \l__stex_refs_post_tl ,
967 }
968 \cs_new_protected:Nn \__stex_refs_args:n {
969   \tl_clear:N \l__stex_refs_linktext_tl
970   \tl_clear:N \l__stex_refs_fallback_tl
971   \tl_clear:N \l__stex_refs_pre_tl
972   \tl_clear:N \l__stex_refs_post_tl
973   \str_clear:N \l__stex_refs_repo_str
974   \keys_set:nn { stex / sref } { #1 }
975 }

```

The actual macro:

```

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

```

```

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

```

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

\srefsym

```

1038 \NewDocumentCommand \srefsym { 0{} m}{
1039   \stex_get_symbol:n { #2 }
1040   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1041 }

```



```

1042
1043 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1044   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1045     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1046   }{
1047     \__stex_refs_args:n { #1 }
1048     \str_if_empty:NTF \l__stex_refs_indocument_str {
1049       \tl_if_exist:cTF{sref_sym_#2 _type}{
1050         % doc uri in \l_tmpb_str
1051         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1052         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1053           % reference
1054           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1055             \cs_if_exist:cTF{autoref}{
1056               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1057             }{
1058               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1059             }
1060           }{
1061             \ltx@ifpackageloaded{hyperref}{
1062               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1063             }{
1064               \l__stex_refs_linktext_tl
1065             }
1066           }
1067         }{
1068           % URL
1069           \ltx@ifpackageloaded{hyperref}{
1070             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1071           }{
1072             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1073           }
1074         }
1075       }{
1076         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1077       }
1078     }{
1079       % TODO
1080     }
1081   }
1082 }

```

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

`\srefsymuri`

```

1083 \cs_new_protected:Npn \srefsymuri #1 #2 {
1084   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1085 }

```

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

```

1086 </package>

```

Chapter 27

STEX -Modules Implementation

```
1087 <*package>
1088
1089 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1090
1091 <@@=stex_modules>
1092
1093     Warnings and error messages
1094 \msg_new:nnn{stex}{error/unknownmodule}{
1095     No~module~#1~found
1096 }
1097 \msg_new:nnn{stex}{error/syntax}{
1098     Syntax~error:~#1
1099 }
1100 \msg_new:nnn{stex}{error/siglanguage}{
1101     Module~#1~declares~signature~#2,~but~does~not~
1102     declare~its~language
1103 }
1104 \msg_new:nnn{stex}{warning/deprecated}{
1105     #1~is~deprecated;~please~use~#2~instead!
1106 }
1107 \msg_new:nnn{stex}{error/conflictingmodules}{
1108     Conflicting~imports~for~module~#1
1109 }
```

`\l_stex_current_module_str` The current module:

```
1109 \str_new:N \l_stex_current_module_str
```

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

`\l_stex_all_modules_seq` Stores all available modules

```
1110 \seq_new:N \l_stex_all_modules_seq
```

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

```

\stex_if_in_module_p:
\stex_if_in_module:TF
1111 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1112   \str_if_empty:NTF \l_stex_current_module_str
1113   \prg_return_false: \prg_return_true:
1114 }

(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
1115 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1116   \prop_if_exist:cTF { c_stex_module_#1_prop }
1117   \prg_return_true: \prg_return_false:
1118 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
1119 \cs_new_protected:Nn \stex_add_to_current_module:n {
1120   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1121 }
1122 \cs_new_protected:Npn \STEXexport {
1123   \begingroup
1124   \newlinechar=-1\relax
1125   \endlinechar=-1\relax
1126   %\catcode'\ = 9\relax
1127   \expandafter\endgroup\__stex_modules_export:n
1128 }
1129 \cs_new_protected:Nn \__stex_modules_export:n {
1130   \ignorespaces #1
1131   \stex_add_to_current_module:n { \ignorespaces #1 }
1132   \stex_smsmode_do:
1133 }
1134 \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
1135 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1136   \str_set:Nx \l_tmpa_str { #1 }
1137   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1138 }

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

```

```

\stex_add_import_to_current_module:n
1139 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1140   \str_set:Nx \l_tmpa_str { #1 }
1141   \exp_args:Nno
1142   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1143     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1144   }
1145 }

```

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

`\stex_collect_imports:n`

```

1146 \cs_new_protected:Nn \stex_collect_imports:n {
1147   \seq_clear:N \l_stex_collect_imports_seq
1148   \__stex_modules_collect_imports:n {#1}
1149 }
1150 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1151   \seq_map_inline:cn {c_stex_module_#1_imports} {
1152     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1153       \__stex_modules_collect_imports:n { ##1 }
1154     }
1155   }
1156   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1157     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1158   }
1159 }

```

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

`\stex_do_up_to_module:n`

```

1160 \int_new:N \l__stex_modules_group_depth_int
1161 \tl_new:N \l__stex_modules_aftergroup_tl
1162 \cs_new_protected:Nn \stex_do_up_to_module:n {
1163   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1164     #1
1165   }{
1166     #1
1167     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1168     \aftergroup\__stex_modules_aftergroup_do:
1169   }
1170 }
1171 \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1172   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1173     \l__stex_modules_aftergroup_tl
1174     \tl_clear:N \l__stex_modules_aftergroup_tl
1175   }{
1176     \l__stex_modules_aftergroup_tl
1177     \aftergroup\__stex_modules_aftergroup_do:
1178   }
1179 }
1180 \cs_new_protected:Nn \stex_reset_up_to_module: {
1181
1182   \tl_gset_eq:NN \l__stex_modules_aftergroup_tl \l__stex_modules_aftergroup_outer_tl
1183 }

```

(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).

1184

(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.

```

1185 \str_new:N \l_stex_modules_ns_str
1186 \str_new:N \l_stex_modules_subpath_str
1187 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1188   \str_set:Nx \l_tmpa_str { #1 }
1189   \seq_set_eq:NN \l_tmpa_seq #2
1190   % split off file extension
1191   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1192   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1193   \seq_get_left:NN \l_tmpb_seq \l_tmppb_str
1194   \seq_put_right:No \l_tmpa_seq \l_tmppb_str
1195
1196   \bool_set_true:N \l_tmpa_bool
1197   \bool_while_do:Nn \l_tmpa_bool {
1198     \seq_pop_left:NN \l_tmpa_seq \l_tmppb_str
1199     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1200       {source} { \bool_set_false:N \l_tmpa_bool }
1201     }{}{
1202       \seq_if_empty:NT \l_tmpa_seq {
1203         \bool_set_false:N \l_tmpa_bool
1204       }
1205     }
1206   }
1207
1208   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1209   \str_if_empty:NTF \l_stex_modules_subpath_str {
1210     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1211   }{
1212     \str_set:Nx \l_stex_modules_ns_str {
1213       \l_tmpa_str/\l_stex_modules_subpath_str
1214     }
1215   }
1216 }
1217
1218 \cs_new_protected:Nn \stex_modules_current_namespace: {
1219   \str_clear:N \l_stex_modules_subpath_str
1220   \prop_if_exist:NTF \l_stex_current_repository_prop {
1221     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1222     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1223   }{
1224     % split off file extension
1225     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1226     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1227     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1228     \seq_get_left:NN \l_tmpb_seq \l_tmppb_str
1229     \seq_put_right:No \l_tmpa_seq \l_tmppb_str
1230     \str_set:Nx \l_stex_modules_ns_str {
1231       file:/\stex_path_to_string:N \l_tmpa_seq
1232     }
1233   }
1234 }

```

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

27.1 The smodule environment

smodule arguments:

```

1235 \keys_define:nn { stex / module } {
1236   title      .tl_set:N      = \smodulename ,
1237   type       .str_set_x:N    = \smodulename ,
1238   id         .str_set_x:N    = \smoduleid ,
1239   deprecate  .str_set_x:N    = \l_stex_module_deprecate_str ,
1240   ns         .str_set_x:N    = \l_stex_module_ns_str ,
1241   lang       .str_set_x:N    = \l_stex_module_lang_str ,
1242   sig        .str_set_x:N    = \l_stex_module_sig_str ,
1243   creators   .str_set_x:N    = \l_stex_module_creators_str ,
1244   contributors .str_set_x:N  = \l_stex_module_contributors_str ,
1245   meta       .str_set_x:N    = \l_stex_module_meta_str ,
1246   srccite    .str_set_x:N    = \l_stex_module_srccite_str
1247 }
1248
1249 \cs_new_protected:Nn \__stex_modules_args:n {
1250   \str_clear:N \smodulename
1251   \str_clear:N \smodulename
1252   \str_clear:N \smoduleid
1253   \str_clear:N \l_stex_module_ns_str
1254   \str_clear:N \l_stex_module_deprecate_str
1255   \str_clear:N \l_stex_module_lang_str
1256   \str_clear:N \l_stex_module_sig_str
1257   \str_clear:N \l_stex_module_creators_str
1258   \str_clear:N \l_stex_module_contributors_str
1259   \str_clear:N \l_stex_module_meta_str
1260   \str_clear:N \l_stex_module_srccite_str
1261   \keys_set:nn { stex / module } { #1 }
1262 }
1263
1264 % module parameters here? In the body?
1265
```

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

```

1266 \cs_new_protected:Nn \stex_module_setup:nn {
1267   \tl_gset_eq:NN \l__stex_modules_aftergroup_outer_tl \l__stex_modules_aftergroup_tl
1268   \tl_clear:N \l__stex_modules_aftergroup_tl
1269   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1270   \str_set:Nx \l_stex_module_name_str { #2 }
1271   \__stex_modules_args:n { #1 }
1272
1273   First, we set up the name and namespace of the module.
1274   Are we in a nested module?
1275
1276   \stex_if_in_module:TF {
1277     % Nested module
1278     \prop_get:cnN {c_stex_module\l_stex_current_module_str_prop}
1279     { ns } \l_stex_module_ns_str
1280     \str_set:Nx \l_stex_module_name_str {
1281       \prop_item:cn {c_stex_module\l_stex_current_module_str_prop}
1282       { name } / \l_stex_module_name_str
1283     }
1284   }
1285 }

```

```

1281 % not nested:
1282 \str_if_empty:NT \l_stex_module_ns_str {
1283   \stex_modules_current_namespace:
1284   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1285   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1286     / {\l_stex_module_ns_str}
1287   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1288   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1289     \str_set:Nx \l_stex_module_ns_str {
1290       \stex_path_to_string:N \l_tmpa_seq
1291     }
1292   }
1293 }
1294 }

```

Next, we determine the language of the module:

```

1295 \str_if_empty:NT \l_stex_module_lang_str {
1296   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1297   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1298   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1299   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1300   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1301     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1302       inferred~from~file~name}
1303     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1304   }
1305 }
1306
1307 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1308   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1309     \l_tmpa_str {
1310       \ltx@ifpackageloaded{babel}{
1311         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1312       }{}
1313     } {
1314       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1315     }
1316   }}

```

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

```

1317 \str_if_empty:NTF \l_stex_module_sig_str {
1318   \exp_args:Nnx \prop_gset_from_keyval:cn {
1319     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1320   } {
1321     name      = \l_stex_module_name_str ,
1322     ns        = \l_stex_module_ns_str ,
1323     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1324     lang      = \l_stex_module_lang_str ,
1325     sig       = \l_stex_module_sig_str ,
1326     deprecate = \l_stex_module_deprecate_str ,
1327     meta      = \l_stex_module_meta_str
1328   }
1329   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}

```

```

1330 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1331 \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _copymodules}
1332 \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1333 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1334 \str_if_empty:NT \l_stex_module_meta_str {
1335   \str_set:Nx \l_stex_module_meta_str {
1336     \c_stex_metatheory_ns_str ? Metatheory
1337   }
1338 }
1339 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1340   \bool_set_true:N \l_stex_in_meta_bool
1341   \exp_args:Nx \stex_add_to_current_module:n {
1342     \bool_set_true:N \l_stex_in_meta_bool
1343     \stex_activate_module:n {\l_stex_module_meta_str}
1344     \bool_set_false:N \l_stex_in_meta_bool
1345   }
1346   \stex_activate_module:n {\l_stex_module_meta_str}
1347   \bool_set_false:N \l_stex_in_meta_bool
1348 }
1349 }{
1350   \str_if_empty:NT \l_stex_module_lang_str {
1351     \msg_error:nnxx{stex}{error/siglanguage}{
1352       \l_stex_module_ns_str?\l_stex_module_name_str
1353     }{\l_stex_module_sig_str}
1354   }
1355
1356   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1357   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1358   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1359   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1360   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1361   \str_set:Nx \l_tmpa_str {
1362     \stex_path_to_string:N \l_tmpa_seq /
1363     \l_tmpa_str . \l_stex_module_sig_str .tex
1364   }
1365   \IfFileExists \l_tmpa_str {
1366     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1367       \str_clear:N \l_stex_current_module_str
1368       \seq_clear:N \l_stex_all_modules_seq
1369       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1370     }
1371   }{
1372     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1373   }
1374   \stex_if_smsmode:F {
1375     \stex_activate_module:n {
1376       \l_stex_module_ns_str ? \l_stex_module_name_str
1377     }
1378   }
1379   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1380 }
1381 \str_if_empty:NF \l_stex_module_deprecate_str {

```



```

1382 \msg_warning:nnxx{stex}{warning/deprecated}{
1383   Module~\l_stex_current_module_str
1384 }{
1385   \l_stex_module_deprecate_str
1386 }
1387 }
1388 \seq_put_right:Nx \l_stex_all_modules_seq {
1389   \l_stex_module_ns_str ? \l_stex_module_name_str
1390 }
1391 }

```

(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}
1392 \cs_new_protected:Nn \__stex_modules_begin_module: {
1393   \stex_reactivate_macro:N \STEXexport
1394   \stex_reactivate_macro:N \importmodule
1395   \stex_reactivate_macro:N \symdecl
1396   \stex_reactivate_macro:N \notation
1397   \stex_reactivate_macro:N \symdef
1398 }
1399 \stex_debug:nn{modules}{
1400   New~module:\\
1401   Namespace::~\l_stex_module_ns_str\\
1402   Name::~\l_stex_module_name_str\\
1403   Language::~\l_stex_module_lang_str\\
1404   Signature::~\l_stex_module_sig_str\\
1405   Metatheory::~\l_stex_module_meta_str\\
1406   File::~\stex_path_to_string:N \g_stex_currentfile_seq
1407 }
1408
1409 \stex_if_smsmode:F{
1410   \begin{stex_annotate_env} {theory} {
1411     \l_stex_module_ns_str ? \l_stex_module_name_str
1412   }
1413
1414   \stex_annotate_invisible:nnn{header}{} {
1415     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1416     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1417     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1418       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1419     }
1420     \str_if_empty:NF \smodulotype {
1421       \stex_annotate:nnn{type}{\smodulotype}{}
1422     }
1423   }
1424 }
1425 % TODO: Inherit metatheory for nested modules?
1426 }
1427 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `__stex_modules_begin_module:.`)

```

1428 \cs_new_protected:Nn \__stex_modules_end_module: {
1429   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}}
1430 }

```

(End definition for __stex_modules_end_module:.)

The core environment

```

1431 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1432 \NewDocumentEnvironment { smodule } { 0 } { m } {
1433   \stex_module_setup:nn{#1}{#2}
1434   \par
1435   \stex_if_smsmode:F{
1436     \tl_clear:N \l_tmpa_tl
1437     \clist_map_inline:Nn \smoduletype {
1438       \tl_if_exist:cT {\__stex_modules_smodule_##1_start:}{
1439         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_start:}}
1440       }
1441     }
1442     \tl_if_empty:NTF \l_tmpa_tl {
1443       \__stex_modules_smodule_start:
1444     }{
1445       \l_tmpa_tl
1446     }
1447   }
1448   \__stex_modules_begin_module:
1449   \str_if_empty:NF \smoduleid {
1450     \stex_ref_new_doc_target:n \smoduleid
1451   }
1452   \stex_smsmode_do:
1453 } {
1454   \__stex_modules_end_module:
1455   \stex_if_smsmode:F {
1456     \end{stex_annotate_env}
1457     \clist_set:No \l_tmpa_clist \smoduletype
1458     \tl_clear:N \l_tmpa_tl
1459     \clist_map_inline:Nn \l_tmpa_clist {
1460       \tl_if_exist:cT {\__stex_modules_smodule_##1_end:}{
1461         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_modules_smodule_##1_end:}}
1462       }
1463     }
1464     \tl_if_empty:NTF \l_tmpa_tl {
1465       \__stex_modules_smodule_end:
1466     }{
1467       \l_tmpa_tl
1468     }
1469   }
1470 }

```

\stexpatchmodule

```

1471 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1472 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1473
1474 \newcommand\stexpatchmodule[3] [] {

```

```

1475 \str_set:Nx \l_tmpa_str{ #1 }
1476 \str_if_empty:NTF \l_tmpa_str {
1477   \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1478   \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1479 }{
1480   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1481   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1482 }
1483 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1484 \NewDocumentCommand \STEXModule { m } {
1485   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1486   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1487   \tl_set:Nn \l_tmpa_tl {
1488     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1489   }
1490   \seq_map_inline:Nn \l_stex_all_modules_seq {
1491     \str_set:Nn \l_tmpb_str { ##1 }
1492     \str_if_eq:eeT { \l_tmpa_str } {
1493       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1494     } {
1495       \seq_map_break:n {
1496         \tl_set:Nn \l_tmpa_tl {
1497           \stex_invoke_module:n { ##1 }
1498         }
1499       }
1500     }
1501   }
1502   \l_tmpa_tl
1503 }
1504
1505 \cs_new_protected:Nn \stex_invoke_module:n {
1506   \stex_debug:nn{modules}{Invoking~module~#1}
1507   \peek_charcode_remove:NTF ! {
1508     \__stex_modules_invoke_uri:nN { #1 }
1509   } {
1510     \peek_charcode_remove:NTF ? {
1511       \__stex_modules_invoke_symbol:nn { #1 }
1512     } {
1513       \msg_error:nnx{stex}{error/syntax}{
1514         ?~or~!~expected~after~
1515         \c_backslash_str STEXModule{#1}
1516       }
1517     }
1518   }
1519 }
1520
1521 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {

```

```

1522 \str_set:Nn #2 { #1 }
1523 }
1524
1525 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1526 \stex_invoke_symbol:n{#1?#2}
1527 }

```

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

`\stex_activate_module:n`

```

1528 \bool_new:N \l_stex_in_meta_bool
1529 \bool_set_false:N \l_stex_in_meta_bool
1530 \cs_new_protected:Nn \stex_activate_module:n {
1531 \stex_debug:nn{modules}{Activating~module~#1}
1532 \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1533 \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1534 }
1535 \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1536 \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1537 \use:c{ c_stex_module_#1_code }
1538 }
1539 }

```

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

```

1540 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1541 ⟨*package⟩
1542
1543 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1544
```

28.1 SMS Mode

```
1545 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1546 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1547 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1548 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1549
1550 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1551   \makeatletter
1552   \makeatother
1553   \ExplSyntaxOn
1554   \ExplSyntaxOff
1555   \rustexBREAK
1556 }
1557
1558 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1559   \symdef
1560   \importmodule
1561   \notation
1562   \symdecl
1563   \STEXexport
1564   \inlineass
1565   \inlinedef
1566   \inlineex
1567   \endinput
1568   \setnotation
```

```

1569 \copynotation
1570 \assign
1571 \renamedekl
1572 \donotcopy
1573 \instantiate
1574 }
1575
1576 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1577   \tl_to_str:n {
1578     smodule,
1579     copymodule,
1580     interpretmodule,
1581     sdefinition,
1582     sexample,
1583     sassertion,
1584     sparagraph,
1585     mathstructure
1586   }
1587 }

```

(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
1588 \bool_new:N \g__stex_smsmode_bool
1589 \bool_set_false:N \g__stex_smsmode_bool
1590 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1591   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1592 }

```

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

```

\_stex_smsmode_in_smsmode:nn
1593 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1594   \vbox_set:Nn \l_tmpa_box {
1595     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1596     \bool_gset_true:N \g__stex_smsmode_bool
1597     #2
1598     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1599   }
1600   \box_clear:N \l_tmpa_box
1601 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

```

\stex_file_in_smsmode:nn
1602 \quark_new:N \q__stex_smsmode_break
1603
1604 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1605   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq { {#1}{#2} }
1606   \stex_smsmode_do:
1607 }
1608
1609 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1610   \stex_filestack_push:n{#1}

```

```

1611 \seq_gclear:N \l__stex_smsmode_importmodules_seq
1612 % ----- new -----
1613 \__stex_smsmode_in_smsmode:nn{#1}{
1614   \let\importmodule\__stex_smsmode_importmodule:
1615   \seq_clear:N \g_stex_smsmode_allowedenvs_seq
1616   \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1617   \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1618   \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1619   \everyeof{\q__stex_smsmode_break\noexpand}
1620   \expandafter\expandafter\expandafter
1621   \stex_smsmode_do:
1622   \csname @ @ input\endcsname "#1"\relax
1623 }
1624 % ----- new -----
1625 \__stex_smsmode_in_smsmode:nn{#1} {
1626   #2
1627   % ----- new -----
1628   \begingroup
1629   \stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1630   \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1631     \stex_import_module_uri:nn ##1
1632     \stex_import_require_module:nnnn
1633     \l_stex_import_ns_str
1634     \l_stex_import_archive_str
1635     \l_stex_import_path_str
1636     \l_stex_import_name_str
1637   }
1638   \endgroup
1639   \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1640   % ----- new -----
1641   \everyeof{\q__stex_smsmode_break\noexpand}
1642   \expandafter\expandafter\expandafter
1643   \stex_smsmode_do:
1644   \csname @ @ input\endcsname "#1"\relax
1645 }
1646 \stex_filestack_pop:
1647 }

```

(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:

```

1648 \cs_new_protected:Npn \stex_smsmode_do: {
1649   \stex_if_smsmode:T {
1650     \__stex_smsmode_do:w
1651   }
1652 }
1653 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1654   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 }}{
1655     \expandafter\if\expandafter\relax\noexpand#1
1656     \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1657   \else\expandafter\__stex_smsmode_do:w\fi
1658 }{
1659   \__stex_smsmode_do:w % #1

```

```

1660 }
1661 }
1662 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1663   \cs_if_eq:NNTF #1 \q__stex_smsmode_break {
1664     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {
1665       #1\__stex_smsmode_do:w
1666     }{
1667       \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1668         #1
1669       }{
1670         \cs_if_eq:NNTF \begin #1 {
1671           \__stex_smsmode_check_begin:n
1672         }{
1673           \cs_if_eq:NNTF \end #1 {
1674             \__stex_smsmode_check_end:n
1675           }{
1676             \__stex_smsmode_do:w
1677           }
1678         }
1679       }
1680     }
1681   }
1682 }
1683
1684 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1685   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1686     \begin{#1}
1687   }{
1688     \__stex_smsmode_do:w
1689   }
1690 }
1691 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1692   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1693     \end{#1}\__stex_smsmode_do:w
1694   }{
1695     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1696   }
1697 }

```

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

28.2 Inheritance

```

1698 <@@=stex_importmodule>

```

`\stex_import_module_uri:nn`

```

1699 \cs_new_protected:Nn \stex_import_module_uri:nn {
1700   \str_set:Nx \l_stex_import_archive_str { #1 }
1701   \str_set:Nn \l_stex_import_path_str { #2 }
1702
1703   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1704   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1705   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1706 }

```



```

1707 \stex_modules_current_namespace:
1708 \bool_lazy_all:nTF {
1709   {\str_if_empty_p:N \l_stex_import_archive_str}
1710   {\str_if_empty_p:N \l_stex_import_path_str}
1711   {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }
1712 }{
1713   \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1714   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1715 }{
1716   \str_if_empty:NT \l_stex_import_archive_str {
1717     \prop_if_exist:NT \l_stex_current_repository_prop {
1718       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1719     }
1720   }
1721   \str_if_empty:NTF \l_stex_import_archive_str {
1722     \str_if_empty:NF \l_stex_import_path_str {
1723       \str_set:Nx \l_stex_import_ns_str {
1724         \l_stex_module_ns_str / \l_stex_import_path_str
1725       }
1726     }
1727   }{
1728     \stex_require_repository:n \l_stex_import_archive_str
1729     \prop_get:cnN { c_stex_mathhub\_l_stex_import_archive_str\_manifest_prop } { ns }
1730     \l_stex_import_ns_str
1731     \str_if_empty:NF \l_stex_import_path_str {
1732       \str_set:Nx \l_stex_import_ns_str {
1733         \l_stex_import_ns_str / \l_stex_import_path_str
1734       }
1735     }
1736   }
1737 }
1738 }

```

(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
1739 \str_new:N \l_stex_import_name_str
1740 \str_new:N \l_stex_import_archive_str
1741 \str_new:N \l_stex_import_path_str
1742 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nnnn {\{ns\}} {\{archive-ID\}} {\{path\}} {\{name\}}
1743 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1744   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1745
1746     % archive
1747     \str_set:Nx \l_tmpa_str { #2 }
1748     \str_if_empty:NTF \l_tmpa_str {
1749       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1750     } {
1751       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1752       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq

```

```

1753     \seq_put_right:Nn \l_tmpa_seq { source }
1754 }
1755
1756 % path
1757 \str_set:Nx \l_tmpb_str { #3 }
1758 \str_if_empty:NTF \l_tmpb_str {
1759     \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1760
1761     \ltx@ifpackageloaded{babel} {
1762         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1763             { \language } \l_tmpb_str {
1764             \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1765         }
1766     } {
1767         \str_clear:N \l_tmpb_str
1768     }
1769
1770     \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1771     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1772         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1773     }{
1774         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1775         \IfFileExists{ \l_tmpa_str.tex }{
1776             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1777         }{
1778             % try english as default
1779             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1780             \IfFileExists{ \l_tmpa_str.en.tex }{
1781                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1782             }{
1783                 \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1784             }
1785         }
1786     }
1787
1788 } {
1789     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1790     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1791
1792     \ltx@ifpackageloaded{babel} {
1793         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1794             { \language } \l_tmpb_str {
1795             \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1796         }
1797     } {
1798         \str_clear:N \l_tmpb_str
1799     }
1800
1801     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1802
1803     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1804     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1805         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1806     }{

```

```

1807 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1808 \IfFileExists{ \l_tmpa_str/#4.tex }{
1809   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1810 }{
1811   % try english as default
1812   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1813   \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1814     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1815   }{
1816     \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1817     \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1818       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1819     }{
1820       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1821       \IfFileExists{ \l_tmpa_str.tex }{
1822         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1823       }{
1824         % try english as default
1825         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1826         \IfFileExists{ \l_tmpa_str.en.tex }{
1827           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1828         }{
1829           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1830         }
1831       }
1832     }
1833   }
1834 }
1835 }
1836 }
1837
1838 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1839   \seq_clear:N \l_stex_all_modules_seq
1840   \str_clear:N \l_stex_current_module_str
1841   \str_set:Nx \l_tmpb_str { #2 }
1842   \str_if_empty:NF \l_tmpb_str {
1843     \stex_set_current_repository:n { #2 }
1844   }
1845   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1846 }
1847
1848 \stex_if_module_exists:nF { #1 ? #4 } {
1849   \msg_error:nnx{stex}{error/unknownmodule}{
1850     #1?#4~(in~file~\g__stex_importmodule_file_str)
1851   }
1852 }
1853 }
1854 \stex_activate_module:n { #1 ? #4 }
1855 }

```

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

`\importmodule`

```

1856 \NewDocumentCommand \importmodule { 0{ } m } {

```

```

1857 \stex_import_module_uri:nn { #1 } { #2 }
1858 \stex_debug:nn{modules}{Importing~module::~
1859   \l_stex_import_ns_str ? \l_stex_import_name_str
1860 }
1861 \stex_import_require_module:nnnn
1862 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1863 { \l_stex_import_path_str } { \l_stex_import_name_str }
1864 \stex_if_smsmode:F {
1865   \stex_annotate_invisible:nnn
1866   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1867 }
1868 \exp_args:Nx \stex_add_to_current_module:n {
1869   \stex_import_require_module:nnnn
1870   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1871   { \l_stex_import_path_str } { \l_stex_import_name_str }
1872 }
1873 \exp_args:Nx \stex_add_import_to_current_module:n {
1874   \l_stex_import_ns_str ? \l_stex_import_name_str
1875 }
1876 \stex_smsmode_do:
1877 \ignorespacesandpars
1878 }
1879 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1880 \NewDocumentCommand \usemodule { 0{} m } {
1881   \stex_if_smsmode:F {
1882     \stex_import_module_uri:nn { #1 } { #2 }
1883     \stex_import_require_module:nnnn
1884     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1885     { \l_stex_import_path_str } { \l_stex_import_name_str }
1886     \stex_annotate_invisible:nnn
1887     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1888   }
1889   \stex_smsmode_do:
1890   \ignorespacesandpars
1891 }

```

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

```

1892 \</package>

```

Chapter 29

STEX -Symbols Implementation

```
1893 <*package>
1894
1895 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1896
      Warnings and error messages
1897 \msg_new:nnn{stex}{error/wrongargs}{
1898   args~value~in~symbol~declaration~for~#1~
1899   needs~to~be~i,~a,~b~or~B,~but~#2~given
1900 }
1901 \msg_new:nnn{stex}{error/unknownsymbol}{
1902   No~symbol~#1~found!
1903 }
1904 \msg_new:nnn{stex}{error/seqlength}{
1905   Expected~#1~arguments;~got~#2!
1906 }
```

29.1 Symbol Declarations

```
1907 <@@=stex_symdecl>

\stex_all_symbols:n Map over all available symbols
1908 \cs_new_protected:Nn \stex_all_symbols:n {
1909   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1910   \seq_map_inline:Nn \l_stex_all_modules_seq {
1911     \seq_map_inline:cn{c_stex_module_##1_constants}{
1912       \__stex_symdecl_all_symbols_cs{##1?####1}
1913     }
1914   }
1915 }

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

\STEXsymbol
1916 \NewDocumentCommand \STEXsymbol { m } {
1917   \stex_get_symbol:n { #1 }
```

```

1918 \exp_args:No
1919 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1920 }

```

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

`symdecl` arguments:

```

1921 \keys_define:nn { stex / symdecl } {
1922   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1923   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1924   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1925   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1926   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1927   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1928   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1929   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1930   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1931   assoc     .choices:nn =
1932     {bin,binl,binr,pre,conj,pwconj}
1933     {\str_set:Nx \l_stex_symdecl_assoc_type_str {\l_keys_choice_tl}}
1934 }
1935
1936 \bool_new:N \l_stex_symdecl_make_macro_bool
1937
1938 \cs_new_protected:Nn \__stex_symdecl_args:n {
1939   \str_clear:N \l_stex_symdecl_name_str
1940   \str_clear:N \l_stex_symdecl_args_str
1941   \str_clear:N \l_stex_symdecl_deprecate_str
1942   \str_clear:N \l_stex_symdecl_assoc_type_str
1943   \bool_set_false:N \l_stex_symdecl_local_bool
1944   \tl_clear:N \l_stex_symdecl_type_tl
1945   \tl_clear:N \l_stex_symdecl_definiens_tl
1946
1947   \keys_set:nn { stex / symdecl } { #1 }
1948 }

```

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

```

1949
1950 \NewDocumentCommand \symdecl { s m O{} } {
1951   \__stex_symdecl_args:n { #3 }
1952   \IfBooleanTF #1 {
1953     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1954   } {
1955     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1956   }
1957   \stex_symdecl_do:n { #2 }
1958   \stex_smsmode_do:
1959 }
1960
1961 \cs_new_protected:Nn \stex_symdecl_do:nn {
1962   \__stex_symdecl_args:n{#1}
1963   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1964   \stex_symdecl_do:n{#2}
1965 }

```

```

1966
1967 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1968 \cs_new_protected:Nn \stex_symdecl_do:n {
1969   \stex_if_in_module:F {
1970     % TODO throw error? some default namespace?
1971   }
1972
1973   \str_if_empty:NT \l_stex_symdecl_name_str {
1974     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1975   }
1976
1977   \prop_if_exist:cT { l_stex_symdecl_
1978     \l_stex_current_module_str ?
1979     \l_stex_symdecl_name_str
1980   }_prop
1981   {
1982     % TODO throw error (beware of circular dependencies)
1983   }
1984
1985   \prop_clear:N \l_tmpa_prop
1986   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1987   \seq_clear:N \l_tmpa_seq
1988   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1989   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1990
1991   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1992     \str_if_empty:NF \l_stex_module_deprecate_str {
1993       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1994     }
1995   }
1996   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1997
1998   \exp_args:No \stex_add_constant_to_current_module:n {
1999     \l_stex_symdecl_name_str
2000   }
2001
2002   % arity/args
2003   \int_zero:N \l_tmpb_int
2004
2005   \bool_set_true:N \l_tmpa_bool
2006   \str_map_inline:Nn \l_stex_symdecl_args_str {
2007     \token_case_meaning:NnF ##1 {
2008       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2009       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2010       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2011       {\tl_to_str:n a} {
2012         \bool_set_false:N \l_tmpa_bool
2013         \int_incr:N \l_tmpb_int
2014       }
2015       {\tl_to_str:n B} {

```

```

2016         \bool_set_false:N \l_tmpa_bool
2017         \int_incr:N \l_tmpb_int
2018     }
2019 }{
2020     \msg_error:nnxx{stex}{error/wrongargs}{
2021         \l_stex_current_module_str ?
2022         \l_stex_symdecl_name_str
2023     }{##1}
2024 }
2025 }
2026 \bool_if:NTF \l_tmpa_bool {
2027     % possibly numeric
2028     \str_if_empty:NTF \l_stex_symdecl_args_str {
2029         \prop_put:Nnn \l_tmpa_prop { args } {}
2030         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2031     }{
2032         \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
2033         \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2034         \str_clear:N \l_tmpa_str
2035         \int_step_inline:nn \l_tmpa_int {
2036             \str_put_right:Nn \l_tmpa_str i
2037         }
2038         \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2039     }
2040 } {
2041     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2042     \prop_put:Nnx \l_tmpa_prop { arity }
2043         { \str_count:N \l_stex_symdecl_args_str }
2044 }
2045 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2046
2047 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2048     \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2049 }{
2050     \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2051 }
2052
2053 % semantic macro
2054
2055 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2056     \exp_args:Nx \stex_do_up_to_module:n {
2057         \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2058             \l_stex_current_module_str ? \l_stex_symdecl_name_str
2059         }}
2060     }
2061
2062     \bool_if:NF \l_stex_symdecl_local_bool {
2063         \exp_args:Nx \stex_add_to_current_module:n {
2064             \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2065                 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2066             } }
2067         }
2068     }
2069 }

```



```

2070
2071 \stex_debug:nn{symbols}{New~symbol:~
2072   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2073   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2074   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2075   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
2076 }
2077
2078 % circular dependencies require this:
2079
2080 \prop_if_exist:cF {
2081   \l_stex_symdecl_
2082   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2083   _prop
2084 } {
2085   \exp_args:Nx \stex_do_up_to_module:n {
2086     \prop_set_from_keyval:cn {
2087       \l_stex_symdecl_
2088       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2089       _prop
2090     } {\prop_to_keyval:N \l_tmpa_prop}
2091     \seq_clear:c {
2092       \l_stex_symdecl_
2093       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2094       _notations
2095     }
2096   }
2097 }
2098
2099 \bool_if:NF \l_stex_symdecl_local_bool {
2100   \exp_args:Nx
2101   \stex_add_to_current_module:n {
2102     \seq_clear:c {
2103       \l_stex_symdecl_
2104       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2105       _notations
2106     }
2107     \prop_set_from_keyval:cn {
2108       \l_stex_symdecl_
2109       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2110       _prop
2111     } {
2112       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2113       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2114       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2115       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2116       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2117       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
2118       defined   = \prop_item:Nn \l_tmpa_prop { defined }   }
2119   }
2120 }
2121 }
2122
2123 \stex_if_smsmode:F {

```

```

2124 % \exp_args:Nx \stex_do_up_to_module:n {
2125 % \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2126 % \l_stex_current_module_str ? \l_stex_symdecl_name_str
2127 % }
2128 % }
2129 \stex_if_do_html:T {
2130 \stex_annotate_invisible:nnn {symdecl} {
2131 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2132 } {
2133 \tl_if_empty:NF \l_stex_symdecl_type_tl {
2134 \stex_annotate_invisible:nnn{type}{\l_stex_symdecl_type_tl$}
2135 }
2136 \stex_annotate_invisible:nnn{args}{\l_stex_symdecl_args_tl$}
2137 \prop_item:Nn \l_tmpa_prop { args }
2138 }
2139 \stex_annotate_invisible:nnn{macroname}{\l_stex_symdecl_macroname_tl$}
2140 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2141 \stex_annotate_invisible:nnn{definiens}{\l_stex_symdecl_definiens_tl$}
2142 }
2143 }
2144 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2145 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str$}
2146 }
2147 }
2148 }
2149 }
2150 }

```

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

`\stex_get_symbol:n`

```

2151 \str_new:N \l_stex_get_symbol_uri_str
2152
2153 \cs_new_protected:Nn \stex_get_symbol:n {
2154 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2155 \tl_set:Nn \l_tmpa_tl { #1 }
2156 \__stex_symdecl_get_symbol_from_cs:
2157 }{
2158 % argument is a string
2159 % is it a command name?
2160 \cs_if_exist:cTF { #1 }{
2161 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2162 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2163 \str_if_empty:NNTF \l_tmpa_str {
2164 \exp_args:Nx \cs_if_eq:NNTF {
2165 \tl_head:N \l_tmpa_tl
2166 } \stex_invoke_symbol:n {
2167 \__stex_symdecl_get_symbol_from_cs:
2168 }{
2169 \__stex_symdecl_get_symbol_from_string:n { #1 }
2170 }
2171 } {
2172 \__stex_symdecl_get_symbol_from_string:n { #1 }
2173 }

```

```

2174     }{
2175         % argument is not a command name
2176         \_stex_symdecl_get_symbol_from_string:n { #1 }
2177         % \l_stex_all_symbols_seq
2178     }
2179 }
2180 \str_if_eq:eeF {
2181     \prop_item:cn {
2182         l_stex_symdecl\_l_stex_get_symbol_uri_str _prop
2183     }{ deprecate }
2184 }{}{
2185     \msg_warning:nnxx{stex}{warning/deprecated}{
2186         Symbol~\l_stex_get_symbol_uri_str
2187     }{
2188         \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop}{ deprecate }
2189     }
2190 }
2191 }
2192
2193 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_string:n {
2194     \tl_set:Nn \l_tmpa_tl {
2195         \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2196     }
2197     \str_set:Nn \l_tmpa_str { #1 }
2198     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2199
2200     \stex_all_symbols:n {
2201         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2202             \seq_map_break:n{\seq_map_break:n{
2203                 \tl_set:Nn \l_tmpa_tl {
2204                     \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2205                 }
2206             }}
2207         }
2208     }
2209
2210     \l_tmpa_tl
2211 }
2212
2213 \cs_new_protected:Nn \_stex_symdecl_get_symbol_from_cs: {
2214     \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2215     { \tl_tail:N \l_tmpa_tl }
2216     \tl_if_single:NTF \l_tmpa_tl {
2217         \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2218             \exp_after:wN \str_set:Nn \exp_after:wN
2219             \l_stex_get_symbol_uri_str \l_tmpa_tl
2220         }{
2221             % TODO
2222             % tail is not a single group
2223         }
2224     }{
2225         % TODO
2226         % tail is not a single group
2227     }

```

2228 }

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

29.2 Notations

2229 `<@@=stex_notation>`

notation arguments:

```
2230 \keys_define:nn { stex / notation } {
2231   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2232   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2233   prec .str_set_x:N = \l__stex_notation_prec_str ,
2234   op .tl_set:N = \l__stex_notation_op_tl ,
2235   primary .bool_set:N = \l__stex_notation_primary_bool ,
2236   primary .default:n = {true} ,
2237   unknown .code:n = \str_set:Nx
2238     \l__stex_notation_variant_str \l_keys_key_str
2239 }
2240
2241 \cs_new_protected:Nn \_stex_notation_args:n {
2242   \str_clear:N \l__stex_notation_lang_str
2243   \str_clear:N \l__stex_notation_variant_str
2244   \str_clear:N \l__stex_notation_prec_str
2245   \tl_clear:N \l__stex_notation_op_tl
2246   \bool_set_false:N \l__stex_notation_primary_bool
2247
2248   \keys_set:nn { stex / notation } { #1 }
2249 }
```

`\notation`

```
2250 \NewDocumentCommand \notation { s m O{}} {
2251   \_stex_notation_args:n { #3 }
2252   \tl_clear:N \l_stex_symdecl_definiens_tl
2253   \stex_get_symbol:n { #2 }
2254   \tl_set:Nn \l_stex_notation_after_do_tl {
2255     \__stex_notation_final:
2256     \IfBooleanTF#1{
2257       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2258     }{}
2259     \stex_smsmode_do:\ignorespacesandpars
2260   }
2261   \stex_notation_do:nnnnn
2262     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { args } }
2263     { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str _prop } { arity } }
2264     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2265     { \l__stex_notation_prec_str }
2266   }
2267   \stex_deactivate_macro:Nn \notation {module~environments}
```

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

`\stex_notation_do:nnnnn`

```
2268 \seq_new:N \l__stex_notation_precedences_seq
```

```

2269 \tl_new:N \l__stex_notation_opprec_tl
2270 \int_new:N \l__stex_notation_currarg_int
2271 \tl_new:N \stex_symbol_after_invokation_tl
2272
2273 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2274   \let\l__stex_current_symbol_str\relax
2275   \seq_clear:N \l__stex_notation_precedences_seq
2276   \tl_clear:N \l__stex_notation_opprec_tl
2277   \str_set:Nx \l__stex_notation_args_str { #1 }
2278   \str_set:Nx \l__stex_notation_arity_str { #2 }
2279   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2280   \str_set:Nx \l__stex_notation_prec_str { #4 }
2281
2282   % precedences
2283   \str_if_empty:NTF \l__stex_notation_prec_str {
2284     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2285       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2286     }{
2287       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2288     }
2289   } {
2290     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2291       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2292       \int_step_inline:nn { \l__stex_notation_arity_str } {
2293         \exp_args:NNo
2294         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2295       }
2296     }{
2297       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2298       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2299         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2300         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2301           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2302             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2303           \seq_map_inline:Nn \l_tmpa_seq {
2304             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2305           }
2306         }
2307       }{
2308         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2309           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2310         }{
2311           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2312         }
2313       }
2314     }
2315   }
2316
2317   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2318   \int_step_inline:nn { \l__stex_notation_arity_str } {
2319     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpb_str {
2320       \exp_args:NNo
2321       \seq_put_right:No \l__stex_notation_precedences_seq {
2322         \l__stex_notation_opprec_tl

```

```

2323     }
2324   }
2325 }
2326 \tl_clear:N \l_stex_notation_dummyargs_tl
2327
2328 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2329   \exp_args:NNe
2330   \cs_set:Npn \l_stex_notation_macrocode_cs {
2331     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2332     { \l__stex_notation_suffix_str }
2333     { \l__stex_notation_opprec_tl }
2334     { \exp_not:n { #5 } }
2335   }
2336   \l_stex_notation_after_do_tl
2337 }{
2338   \str_if_in:NnTF \l__stex_notation_args_str b {
2339     \exp_args:Nne \use:nn
2340     {
2341       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2342       \cs_set:Npn \l__stex_notation_arity_str } { {
2343         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2344         { \l__stex_notation_suffix_str }
2345         { \l__stex_notation_opprec_tl }
2346         { \exp_not:n { #5 } }
2347       }}
2348   }{
2349     \str_if_in:NnTF \l__stex_notation_args_str B {
2350       \exp_args:Nne \use:nn
2351       {
2352         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2353         \cs_set:Npn \l__stex_notation_arity_str } { {
2354           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2355           { \l__stex_notation_suffix_str }
2356           { \l__stex_notation_opprec_tl }
2357           { \exp_not:n { #5 } }
2358         } }
2359     }{
2360       \exp_args:Nne \use:nn
2361       {
2362         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2363         \cs_set:Npn \l__stex_notation_arity_str } { {
2364           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2365           { \l__stex_notation_suffix_str }
2366           { \l__stex_notation_opprec_tl }
2367           { \exp_not:n { #5 } }
2368         } }
2369     }
2370   }
2371
2372   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2373   \int_zero:N \l__stex_notation_currarg_int
2374   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2375   \__stex_notation_arguments:
2376 }

```

2377 }

(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

```

2378 \cs_new_protected:Nn \_stex_notation_arguments: {
2379   \int_incr:N \l__stex_notation_currarg_int
2380   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2381     \l_stex_notation_after_do_tl
2382   }{
2383     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2384     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2385     \str_if_eq:NnTF \l_tmpa_str a {
2386       \_stex_notation_argument_assoc:nn{a}
2387     }{
2388       \str_if_eq:NnTF \l_tmpa_str B {
2389         \_stex_notation_argument_assoc:nn{B}
2390       }{
2391         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpb_str
2392         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2393           { \_stex_term_math_arg:nnn
2394             { \l_tmpa_str\int_use:N \l__stex_notation_currarg_int }
2395             { \l_tmpb_str }
2396             { ####\int_use:N \l__stex_notation_currarg_int }
2397           }
2398         }
2399         \_stex_notation_arguments:
2400       }
2401     }
2402   }
2403 }
```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:nn

```

2404 \cs_new_protected:Nn \_stex_notation_argument_assoc:nn {
2405
2406   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2407     {\l__stex_notation_arity_str}{
2408       #2
2409     }
2410   \int_zero:N \l_tmpa_int
2411   \tl_clear:N \l_tmpa_tl
2412   \str_map_inline:Nn \l__stex_notation_args_str {
2413     \int_incr:N \l_tmpa_int
2414     \tl_put_right:Nx \l_tmpa_tl {
2415       \str_if_eq:nnTF {##1}{a}{ {} }{
2416         \str_if_eq:nnTF {##1}{B}{ {} }{
2417           {\_stex_term_arg:nn{##1\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa
2418         }
2419       }
2420     }
2421   }
2422   \exp_after:wN\exp_after:wN\exp_after:wN \def
```

```

2423 \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2424 \exp_after:wN\exp_after:wN\exp_after:wN ##
2425 \exp_after:wN\exp_after:wN\exp_after:wN 1
2426 \exp_after:wN\exp_after:wN\exp_after:wN ##
2427 \exp_after:wN\exp_after:wN\exp_after:wN 2
2428 \exp_after:wN\exp_after:wN\exp_after:wN {
2429   \exp_after:wN \exp_after:wN \exp_after:wN
2430   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2431     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2432   }
2433 }
2434
2435 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2436 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2437   \stex_term_math_assoc_arg:nnnn
2438   { #1\int_use:N \l__stex_notation_currarg_int }
2439   { \l_tmpa_str }
2440   { ####\int_use:N \l__stex_notation_currarg_int }
2441   { \l_tmpa_cs {####1} {####2} }
2442 } }
2443 \__stex_notation_arguments:
2444 }

```

(End definition for __stex_notation_argument_assoc:nn.)

__stex_notation_final: Called after processing all notation arguments

```

2445 \cs_new_protected:Nn \__stex_notation_final: {
2446 % \exp_args:Nne \use:nn
2447 % {
2448 % \cs_generate_from_arg_count:cNnn {
2449 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2450 %   \l__stex_notation_suffix_str
2451 %   _cs
2452 % }
2453 % \cs_set:Npn \l__stex_notation_arity_str } { {
2454 %   \exp_after:wN \exp_after:wN \exp_after:wN
2455 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2456 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sym
2457 % } }
2458
2459 % \tl_if_empty:NF \l__stex_notation_op_tl {
2460 %   \cs_set:cpx {
2461 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2462 %     \l__stex_notation_suffix_str
2463 %     _cs
2464 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2465 % }
2466
2467 \exp_args:Nx \stex_do_up_to_module:n {
2468   \cs_generate_from_arg_count:cNnn {
2469     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2470     \l__stex_notation_suffix_str
2471     _cs
2472   } \cs_set:Npn {\l__stex_notation_arity_str} {

```



```

2473         \exp_after:wN \exp_after:wN \exp_after:wN
2474         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2475         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2476     }
2477     \tl_if_empty:NF \l__stex_notation_op_tl {
2478         \cs_set:cpn {
2479             stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2480             \l__stex_notation_suffix_str
2481             _cs
2482         } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2483     }
2484 }
2485
2486 \exp_args:Ne
2487 \stex_add_to_current_module:n {
2488     \cs_generate_from_arg_count:cNnn {
2489         stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2490         \l__stex_notation_suffix_str
2491         _cs
2492     } \cs_set:Npn {\l__stex_notation_arity_str} {
2493         \exp_after:wN \exp_after:wN \exp_after:wN
2494         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2495         { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2496     }
2497     \tl_if_empty:NF \l__stex_notation_op_tl {
2498         \cs_set:cpn {
2499             stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2500             \l__stex_notation_suffix_str
2501             _cs
2502         } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2503     }
2504 }
2505
2506 \stex_debug:nn{symbols}{
2507     Notation~\l__stex_notation_suffix_str
2508     ~for~\l_stex_get_symbol_uri_str^^J
2509     Operator~precedence:~\l__stex_notation_opprec_tl^^J
2510     Argument~precedences:~
2511     \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2512     Notation: \cs_meaning:c {
2513         stex_notation_\l_stex_get_symbol_uri_str \c_hash_str
2514         \l__stex_notation_suffix_str
2515         _cs
2516     }
2517 }
2518
2519 \exp_args:Nx
2520 \stex_do_up_to_module:n {
2521     \seq_put_right:cx {
2522         l_stex_symdecl_\l_stex_get_symbol_uri_str
2523         _notations
2524     } {
2525         \l__stex_notation_suffix_str
2526     }

```

```

2527 }
2528 \exp_args:Ne
2529 \stex_add_to_current_module:n {
2530   \seq_put_right:cn {
2531     l_stex_symdecl_l_stex_get_symbol_uri_str
2532     _notations
2533   } { \l__stex_notation_suffix_str }
2534 }
2535
2536 \stex_if_smsmode:F {
2537
2538   % HTML annotations
2539   \stex_if_do_html:T {
2540     \stex_annotate_invisible:nnn { notation }
2541     { \l_stex_get_symbol_uri_str } {
2542       \stex_annotate_invisible:nnn { notationfragment }
2543       { \l__stex_notation_suffix_str }{}
2544       \stex_annotate_invisible:nnn { precedence }
2545       { \l__stex_notation_prec_str }{}
2546
2547       \int_zero:N \l_tmpa_int
2548       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2549       \tl_clear:N \l_tmpa_tl
2550       \int_step_inline:nn { \l__stex_notation_arity_str }{
2551         \int_incr:N \l_tmpa_int
2552         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2553         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2554         \str_if_eq:VnTF \l_tmpb_str a {
2555           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2556             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2557             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2558           } }
2559         }{
2560           \str_if_eq:VnTF \l_tmpb_str B {
2561             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2562               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{} ,
2563               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{}
2564             } }
2565           }{
2566             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2567               \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{}
2568             } }
2569           }
2570         }
2571       }
2572       \stex_annotate_invisible:nnn { notationcomp }{}{
2573         \str_set:Nx \l_stex_current_symbol_str {\l_stex_get_symbol_uri_str }
2574         $ \exp_args:Nno \use:nn { \use:c {
2575           stex_notation_ \l_stex_current_symbol_str
2576           \c_hash_str \l__stex_notation_suffix_str _cs
2577         } } { \l_tmpa_tl } $
2578       }
2579     }
2580   }

```

```

2581 }
2582 }

```

(End definition for _stex_notation_final:.)

\setnotation

```

2583 \keys_define:nn { stex / setnotation } {
2584   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2585   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2586   unknown   .code:n      = \str_set:Nx
2587             \l__stex_notation_variant_str \l_keys_key_str
2588 }
2589
2590 \cs_new_protected:Nn \stex_setnotation_args:n {
2591   \str_clear:N \l__stex_notation_lang_str
2592   \str_clear:N \l__stex_notation_variant_str
2593   \keys_set:nn { stex / setnotation } { #1 }
2594 }
2595
2596 \cs_new_protected:Nn \stex_setnotation:n {
2597   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2598   { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2599     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2600     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2601     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2602     { \c_hash_str }
2603     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2604     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2605     \exp_args:Nx \stex_add_to_current_module:n {
2606       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2607       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2608       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2609       { \c_hash_str }
2610       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2611       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2612     }
2613     \stex_debug:nn {notations}{
2614       Setting~default~notation~
2615       {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2616       #1 \\
2617       \expandafter\meaning\csname
2618       l_stex_symdecl_#1 _notations\endcsname
2619     }
2620   }{
2621     % todo throw error
2622   }
2623 }
2624
2625 \NewDocumentCommand \setnotation {m m} {
2626   \stex_get_symbol:n { #1 }
2627   \stex_setnotation_args:n { #2 }
2628   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2629   \stex_smsmode_do:\ignorespacesandpars
2630 }

```

```

2631
2632 \cs_new_protected:Nn \stex_copy_notations:nn {
2633   \stex_debug:nn {notations}{
2634     Copying~notations~from~#2~to~#1\\
2635     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}
2636   }
2637   \tl_clear:N \l_tmpa_tl
2638   \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2639     \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2640   }
2641   \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2642     \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2643     \edef \l_tmpa_tl {
2644       \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2645       \exp_after:wN\exp_after:wN\exp_after:wN \exp_after:wN {
2646         \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2647       }
2648     }
2649     \exp_args:Nx
2650     \stex_do_up_to_module:n {
2651       \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2652       \cs_generate_from_arg_count:cNnn {
2653         stex_notation_ #1 \c_hash_str ##1 _cs
2654       } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2655         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2656       }
2657     }
2658   }
2659 }
2660
2661 \NewDocumentCommand \copynotation {m m} {
2662   \stex_get_symbol:n { #1 }
2663   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2664   \stex_get_symbol:n { #2 }
2665   \exp_args:Noo
2666   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2667   \exp_args:Nx \stex_add_import_to_current_module:n{
2668     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2669   }
2670   \stex_smsmode_do:\ignorespacesandpars
2671 }
2672

```

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

\symdef

```

2673 \keys_define:nn { stex / symdef } {
2674   name .str_set_x:N = \l_stex_symdecl_name_str ,
2675   local .bool_set:N = \l_stex_symdecl_local_bool ,
2676   args .str_set_x:N = \l_stex_symdecl_args_str ,
2677   type .tl_set:N = \l_stex_symdecl_type_tl ,
2678   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2679   op .tl_set:N = \l__stex_notation_op_tl ,
2680   lang .str_set_x:N = \l__stex_notation_lang_str ,

```

```

2681   variant .str_set_x:N = \l__stex_notation_variant_str ,
2682   prec    .str_set_x:N = \l__stex_notation_prec_str ,
2683   assoc   .choices:nn =
2684     {bin,binl,binr,pre,conj,pwconj}
2685     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}},
2686   unknown .code:n      = \str_set:Nx
2687     \l__stex_notation_variant_str \l_keys_key_str
2688 }
2689
2690 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2691   \str_clear:N \l_stex_symdecl_name_str
2692   \str_clear:N \l_stex_symdecl_args_str
2693   \str_clear:N \l_stex_symdecl_astype_str
2694   \bool_set_false:N \l_stex_symdecl_local_bool
2695   \tl_clear:N \l_stex_symdecl_type_tl
2696   \tl_clear:N \l_stex_symdecl_definiens_tl
2697   \str_clear:N \l__stex_notation_lang_str
2698   \str_clear:N \l__stex_notation_variant_str
2699   \str_clear:N \l__stex_notation_prec_str
2700   \tl_clear:N \l__stex_notation_op_tl
2701
2702   \keys_set:nn { stex / symdef } { #1 }
2703 }
2704
2705 \NewDocumentCommand \symdef { m O{} } {
2706   \__stex_notation_symdef_args:n { #2 }
2707   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2708   \stex_symdecl_do:n { #1 }
2709   \tl_set:Nn \l_stex_notation_after_do_tl {
2710     \__stex_notation_final:
2711     \stex_smsmode_do:\ignorespacesandpars
2712   }
2713   \str_set:Nx \l_stex_get_symbol_uri_str {
2714     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2715   }
2716   \exp_args:Nx \stex_notation_do:nnnnn
2717     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { args } }
2718     { \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str_prop} { arity } }
2719     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2720     { \l__stex_notation_prec_str }
2721 }
2722 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

29.3 Variables

```

2723 <@@=stex_variables>
2724
2725 \keys_define:nn { stex / vardef } {
2726   name .str_set_x:N = \l__stex_variables_name_str ,
2727   args .str_set_x:N = \l__stex_variables_args_str ,
2728   type .tl_set:N    = \l__stex_variables_type_tl ,
2729   def .tl_set:N     = \l__stex_variables_def_tl ,

```

```

2730 op      .tl_set:N      = \l__stex_variables_op_tl ,
2731 prec    .str_set_x:N    = \l__stex_variables_prec_str ,
2732 assoc   .choices:nn     =
2733         {bin,binl,binr,pre,conj,pwconj}
2734         {\str_set:Nx \l__stex_variables_assoctype_str {\l_keys_choice_tl}},
2735 bind    .choices:nn     =
2736         {forall,exists}
2737         {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2738 }
2739
2740 \cs_new_protected:Nn \l__stex_variables_args:n {
2741   \str_clear:N \l__stex_variables_name_str
2742   \str_clear:N \l__stex_variables_args_str
2743   \str_clear:N \l__stex_variables_prec_str
2744   \str_clear:N \l__stex_variables_assoctype_str
2745   \str_clear:N \l__stex_variables_bind_str
2746   \tl_clear:N \l__stex_variables_type_tl
2747   \tl_clear:N \l__stex_variables_def_tl
2748   \tl_clear:N \l__stex_variables_op_tl
2749
2750   \keys_set:nn { stex / vardef } { #1 }
2751 }
2752
2753 \NewDocumentCommand \l__stex_variables_do_simple:nnn { m O{} } {
2754   \l__stex_variables_args:n {#2}
2755   \str_if_empty:NT \l__stex_variables_name_str {
2756     \str_set:Nx \l__stex_variables_name_str { #1 }
2757   }
2758   \prop_clear:N \l_tmpa_prop
2759   \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2760
2761   \int_zero:N \l_tmpb_int
2762   \bool_set_true:N \l_tmpa_bool
2763   \str_map_inline:Nn \l__stex_variables_args_str {
2764     \token_case_meaning:NnF ##1 {
2765       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2766       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2767       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2768       {\tl_to_str:n a} {
2769         \bool_set_false:N \l_tmpa_bool
2770         \int_incr:N \l_tmpb_int
2771       }
2772       {\tl_to_str:n B} {
2773         \bool_set_false:N \l_tmpa_bool
2774         \int_incr:N \l_tmpb_int
2775       }
2776     }{
2777       \msg_error:nnxx{stex}{error/wrongargs}{
2778         variable~\l__stex_variables_name_str
2779       }{##1}
2780     }
2781   }
2782   \bool_if:NTF \l_tmpa_bool {
2783     % possibly numeric

```

```

2784 \str_if_empty:NTF \l__stex_variables_args_str {
2785   \prop_put:Nnn \l_tmpa_prop { args } {}
2786   \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2787 }{
2788   \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }
2789   \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2790   \str_clear:N \l_tmpa_str
2791   \int_step_inline:nn \l_tmpa_int {
2792     \str_put_right:Nn \l_tmpa_str i
2793   }
2794   \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2795   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2796 }
2797 } {
2798   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2799   \prop_put:Nnx \l_tmpa_prop { arity }
2800     { \str_count:N \l__stex_variables_args_str }
2801 }
2802 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
2803 \tl_set:cx { #1 }{ \stex_invoke_variable:n { \l__stex_variables_name_str } }
2804
2805 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2806
2807 \tl_if_empty:NF \l__stex_variables_op_tl {
2808   \cs_set:cpx {
2809     stex_var_op_notation_ \l__stex_variables_name_str _cs
2810   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2811 }
2812
2813 \tl_set:Nn \l_stex_notation_after_do_tl {
2814   \exp_args:Nne \use:nn {
2815     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str _cs }
2816     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2817   } {}
2818   \exp_after:wN \exp_after:wN \exp_after:wN
2819   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2820   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2821 }
2822 \stex_if_do_html:T {
2823   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2824     \stex_annotate_invisible:nnn { precedence }
2825       { \l__stex_variables_prec_str }{}
2826   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2827   \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2828   \stex_annotate_invisible:nnn{macroname}{#1}{
2829   \tl_if_empty:NF \l__stex_variables_def_tl {
2830     \stex_annotate_invisible:nnn{definens}{
2831       {\l__stex_variables_def_tl$}
2832     }
2833   \str_if_empty:NF \l__stex_variables_assoctype_str {
2834     \stex_annotate_invisible:nnn{assoctype}{\l__stex_variables_assoctype_str}{
2835   }
2836   \str_if_empty:NF \l__stex_variables_bind_str {
2837     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{

```

```

2838     }
2839     \int_zero:N \l_tmpa_int
2840     \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2841     \tl_clear:N \l_tmpa_tl
2842     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2843       \int_incr:N \l_tmpa_int
2844       \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2845       \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables
2846       \str_if_eq:VnTF \l_tmpb_str a {
2847         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2848           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2849           \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2850         } }
2851       }{
2852         \str_if_eq:VnTF \l_tmpb_str B {
2853           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2854             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int a}{ } ,
2855             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int b}{ }
2856           } }
2857         }{
2858           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2859             \stex_annotate:nnn{argmarker}{\int_use:N \l_tmpa_int}{ }
2860           } }
2861         }
2862       }
2863     }
2864     \stex_annotate_invisible:nnn { notationcomp }{ }{
2865       \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2866       $ \exp_args:Nno \use:nn { \use:c {
2867         stex_var_notation_\l__stex_variables_name_str _cs
2868       } } { \l_tmpa_tl } $
2869     }
2870   }
2871   }\ignorespacesandpars
2872 }
2873
2874 \stex_notation_do:nnnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2875 }
2876
2877 \cs_new:Nn \_stex_reset:N {
2878   \tl_if_exist:NTF #1 {
2879     \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2880   }{
2881     \let \exp_not:N #1 \exp_not:N \undefined
2882   }
2883 }
2884
2885 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2886   \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2887   \exp_args:Nnx \use:nn {
2888     % TODO
2889     \stex_annotate_invisible:nnn {vardecl}{\clist_use:Nn\l__stex_variables_names,}{
2890       #2
2891     }

```



```

2892 }{
2893   \_stex_reset:N \varnot
2894   \_stex_reset:N \vartype
2895   \_stex_reset:N \vardefi
2896 }
2897 }
2898
2899 \NewDocumentCommand \vardef { s } {
2900   \IfBooleanTF#1 {
2901     \__stex_variables_do_complex:nn
2902   }{
2903     \__stex_variables_do_simple:nnn
2904   }
2905 }
2906
2907 \NewDocumentCommand \svar { 0{} m }{
2908   \tl_if_empty:nTF {#1}{
2909     \str_set:Nn \l_tmpa_str { #2 }
2910   }{
2911     \str_set:Nn \l_tmpa_str { #1 }
2912   }
2913   \_stex_term_omv:nn {
2914     var://\l_tmpa_str
2915   }{
2916     \exp_args:Nnx \use:nn {
2917       \def\comp{\_varcomp}
2918       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2919       \comp{ #2 }
2920     }{
2921       \_stex_reset:N \comp
2922       \_stex_reset:N \l_stex_current_symbol_str
2923     }
2924   }
2925 }
2926
2927
2928
2929 \keys_define:nn { stex / varseq } {
2930   name .str_set_x:N = \l__stex_variables_name_str ,
2931   args .int_set:N = \l__stex_variables_args_int ,
2932   type .tl_set:N = \l__stex_variables_type_tl ,
2933   mid .tl_set:N = \l__stex_variables_mid_tl ,
2934   bind .choices:nn =
2935     {forall,exists}
2936     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2937 }
2938
2939 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2940   \str_clear:N \l__stex_variables_name_str
2941   \int_set:Nn \l__stex_variables_args_int 1
2942   \tl_clear:N \l__stex_variables_type_tl
2943   \str_clear:N \l__stex_variables_bind_str
2944
2945   \keys_set:nn { stex / varseq } { #1 }

```

```

2946 }
2947
2948 \NewDocumentCommand \varseq {m O{} m m m}{
2949   \__stex_variables_seq_args:n { #2 }
2950   \str_if_empty:NT \l__stex_variables_name_str {
2951     \str_set:Nx \l__stex_variables_name_str { #1 }
2952   }
2953   \prop_clear:N \l_tmpa_prop
2954   \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2955
2956   \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2957   \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2958     \msg_error:nnxx{stex}{error/seqlength}
2959     {\int_use:N \l__stex_variables_args_int}
2960     {\seq_count:N \l_tmpa_seq}
2961   }
2962   \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2963   \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2964     \msg_error:nnxx{stex}{error/seqlength}
2965     {\int_use:N \l__stex_variables_args_int}
2966     {\seq_count:N \l_tmpb_seq}
2967   }
2968   \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2969   \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2970
2971   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2972   \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
2973
2974   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2975   \int_step_inline:nn \l__stex_variables_args_int {
2976     \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2977   }
2978   \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2979   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2980   \tl_if_empty:NF \l__stex_variables_mid_tl {
2981     \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2982     \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2983   }
2984   \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2985   \int_step_inline:nn \l__stex_variables_args_int {
2986     \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2987   }
2988   \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2989   \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2990
2991
2992   \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2993
2994   \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2995
2996   \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2997
2998   \int_step_inline:nn \l__stex_variables_args_int {
2999     \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {

```

```

3000     \stex_term_math_arg:nnn{i##1}{0}{\exp_not:n{####}##1}
3001   }}
3002 }
3003
3004 \tl_set:Nx \l_tmpa_tl {
3005   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str}{0}{
3006     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
3007   }
3008 }
3009
3010 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
3011
3012 \exp_args:Nno \use:nn {
3013   \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
3014   \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
3015
3016 \stex_debug:nn{sequences}{New~Sequence:~
3017   \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
3018   \prop_to_keyval:N \l_tmpa_prop
3019 }
3020 \stex_if_do_html:T{\stex_annotate_invisible:nnn{varseq}{\l__stex_variables_name_str}{
3021   \tl_if_empty:NF \l__stex_variables_type_tl {
3022     \stex_annotate:nnn {type}{\{$\seqtype\l__stex_variables_type_tl$\}
3023   }
3024   \stex_annotate:nnn {args}{\int_use:N \l__stex_variables_args_int}{\}
3025   \str_if_empty:NF \l__stex_variables_bind_str {
3026     \stex_annotate:nnn {bindtype}{\l__stex_variables_bind_str}{\}
3027   }
3028 }}
3029
3030 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
3031 \ignorespacesandpars
3032 }
3033
3034 </package>

```

Chapter 30

STEX -Terms Implementation

```
3035 <*package>
3036
3037 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
3038
3039 <@@=stex_terms>
3040
3041   Warnings and error messages
3042 \msg_new:nnn{stex}{error/nonotation}{
3043   Symbol~#1~invoked,~but~has~no~notation#2!
3044 }
3045 \msg_new:nnn{stex}{error/notationarg}{
3046   Error~in~parsing~notation~#1
3047 }
3048 \msg_new:nnn{stex}{error/noop}{
3049   Symbol~#1~has~no~operator~notation~for~notation~#2
3050 }
3051 \msg_new:nnn{stex}{error/notallowed}{
3052   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3053 }
3054 \msg_new:nnn{stex}{error/doubleargument}{
3055   Argument~#1~of~symbol~#2~already~assigned
3056 }
3057 \msg_new:nnn{stex}{error/overarity}{
3058   Argument~#1~invalid~for~symbol~#2~with~arity~#3
3059 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3059
3060
3061 \bool_new:N \l_stex_allow_semantic_bool
3062 \bool_set_true:N \l_stex_allow_semantic_bool
3063
```

```

3064 \cs_new_protected:Nn \stex_invoke_symbol:n {
3065   \bool_if:NTF \l_stex_allow_semantic_bool {
3066     \str_if_eq:eeF {
3067       \prop_item:cn {
3068         l_stex_symdecl_#1_prop
3069       }{ deprecate }
3070     }{}{
3071       \msg_warning:nxxx{stex}{warning/deprecated}{
3072         Symbol~#1
3073       }{
3074         \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3075       }
3076     }
3077     \if_mode_math:
3078       \exp_after:wN \__stex_terms_invoke_math:n
3079     \else:
3080       \exp_after:wN \__stex_terms_invoke_text:n
3081     \fi: { #1 }
3082   }{
3083     \msg_error:nxxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3084   }
3085 }
3086
3087 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3088   \peek_charcode_remove:NTF ! {
3089     \__stex_terms_invoke_op_custom:nn {#1}
3090   }{
3091     \__stex_terms_invoke_custom:nn {#1}
3092   }
3093 }
3094
3095 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3096   \peek_charcode_remove:NTF ! {
3097     % operator
3098     \peek_charcode_remove:NTF * {
3099       % custom op
3100       \__stex_terms_invoke_op_custom:nn {#1}
3101     }{
3102       % op notation
3103       \peek_charcode:NTF [ {
3104         \__stex_terms_invoke_op_notation:nw {#1}
3105       }{
3106         \__stex_terms_invoke_op_notation:nw {#1}[]
3107       }
3108     }
3109   }{
3110     \peek_charcode_remove:NTF * {
3111       \__stex_terms_invoke_custom:nn {#1}
3112       % custom
3113     }{
3114       % normal
3115       \peek_charcode:NTF [ {
3116         \__stex_terms_invoke_notation:nw {#1}
3117       }{

```

```

3118         \_stex_terms_invoke_notation:nw {#1}[]
3119     }
3120 }
3121 }
3122 }
3123
3124
3125 \cs_new_protected:Nn \_stex_terms_invoke_op_custom:nn {
3126     \exp_args:Nnx \use:nn {
3127         \def\comp{\_comp}
3128         \str_set:Nn \l_stex_current_symbol_str { #1 }
3129         \bool_set_false:N \l_stex_allow_semantic_bool
3130         \_stex_term_oms:nnn {#1}{#1 \c_hash_str CUSTOM-}{
3131             \comp{ #2 }
3132         }
3133     }{
3134         \_stex_reset:N \comp
3135         \_stex_reset:N \l_stex_current_symbol_str
3136         \bool_set_true:N \l_stex_allow_semantic_bool
3137     }
3138 }
3139
3140 \keys_define:nn { stex / terms } {
3141     lang .tl_set_x:N = \l_stex_notation_lang_str ,
3142     variant .tl_set_x:N = \l_stex_notation_variant_str ,
3143     unknown .code:n = \str_set:Nx
3144         \l_stex_notation_variant_str \l_keys_key_str
3145 }
3146
3147 \cs_new_protected:Nn \_stex_terms_args:n {
3148     \str_clear:N \l_stex_notation_lang_str
3149     \str_clear:N \l_stex_notation_variant_str
3150
3151     \keys_set:nn { stex / terms } { #1 }
3152 }
3153
3154 \cs_new_protected:Nn \stex_find_notation:nn {
3155     \_stex_terms_args:n { #2 }
3156     \seq_if_empty:cTF {
3157         l_stex_symdecl_ #1 _notations
3158     } {
3159         \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3160     } {
3161         \bool_lazy_all:nTF {
3162             {\str_if_empty_p:N \l_stex_notation_variant_str}
3163             {\str_if_empty_p:N \l_stex_notation_lang_str}
3164         }{
3165             \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3166         }{
3167             \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3168                 \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3169             }{
3170                 \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3171                 }{

```

```

3172         \msg_error:nxxx{stex}{error/nonotation}{#1}{
3173             ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3174         }
3175     }
3176 }
3177 }
3178 }
3179
3180 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3181     \exp_args:Nnx \use:nn {
3182         \def\comp{\_comp}
3183         \str_set:Nn \l_stex_current_symbol_str { #1 }
3184         \stex_find_notation:nn { #1 }{ #2 }
3185         \bool_set_false:N \l_stex_allow_semantic_bool
3186         \cs_if_exist:cTF {
3187             stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3188         }{
3189             \_stex_term_oms:nnn { #1 }{
3190                 #1 \c_hash_str \l_stex_notation_variant_str
3191             }{
3192                 \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3193             }
3194         }{
3195             \int_compare:nNnTF {\prop_item:cn {\l_stex_symdecl_#1_prop}{arity}} = 0{
3196                 \cs_if_exist:cTF {
3197                     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3198                 }{
3199                     \tl_set:Nx \stex_symbol_after_invokation_tl {
3200                         \_stex_reset:N \comp
3201                         \_stex_reset:N \stex_symbol_after_invokation_tl
3202                         \_stex_reset:N \l_stex_current_symbol_str
3203                         \bool_set_true:N \l_stex_allow_semantic_bool
3204                     }
3205                     \def\comp{\_comp}
3206                     \str_set:Nn \l_stex_current_symbol_str { #1 }
3207                     \bool_set_false:N \l_stex_allow_semantic_bool
3208                     \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3209                 }{
3210                     \msg_error:nxxx{stex}{error/nonotation}{#1}{
3211                         ~\l_stex_notation_variant_str
3212                     }
3213                 }
3214             }{
3215                 \msg_error:nxxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3216             }
3217         }
3218     }{
3219         \_stex_reset:N \comp
3220         \_stex_reset:N \l_stex_current_symbol_str
3221         \bool_set_true:N \l_stex_allow_semantic_bool
3222     }
3223 }
3224
3225 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {

```

```

3226 \stex_find_notation:nn { #1 }{ #2 }
3227 \cs_if_exist:cTF {
3228   stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3229 }{
3230   \tl_set:Nx \stex_symbol_after_invokation_tl {
3231     \_stex_reset:N \comp
3232     \_stex_reset:N \stex_symbol_after_invokation_tl
3233     \_stex_reset:N \l_stex_current_symbol_str
3234     \bool_set_true:N \l_stex_allow_semantic_bool
3235   }
3236   \def\comp{\_comp}
3237   \str_set:Nn \l_stex_current_symbol_str { #1 }
3238   \bool_set_false:N \l_stex_allow_semantic_bool
3239   \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3240 }{
3241   \msg_error:nxxx{stex}{error/nonotation}{#1}{
3242     ~\l_stex_notation_variant_str
3243   }
3244 }
3245 }
3246
3247 \prop_new:N \l__stex_terms_custom_args_prop
3248
3249 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3250   \exp_args:Nnx \use:nn {
3251     \bool_set_false:N \l_stex_allow_semantic_bool
3252     \def\comp{\_comp}
3253     \str_set:Nn \l_stex_current_symbol_str { #1 }
3254     \prop_clear:N \l__stex_terms_custom_args_prop
3255     \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3256     \prop_get:cnN {
3257       l_stex_symdecl_#1 _prop
3258     }{ args } \l_tmpa_str
3259     \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3260     \tl_set:Nn \arg { \__stex_terms_arg: }
3261     \str_if_empty:NTF \l_tmpa_str {
3262       \_stex_term_oms:nnn {#1}{#1\c_hash_str CUSTOM-}{#2}
3263     }{
3264       \str_if_in:NnTF \l_tmpa_str b {
3265         \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3266       }{
3267         \str_if_in:NnTF \l_tmpa_str B {
3268           \_stex_term_ombind:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3269         }{
3270           \_stex_term_oma:nnn {#1}{#1\c_hash_str CUSTOM-\l_tmpa_str}{#2}
3271         }
3272       }
3273     }
3274     % TODO check that all arguments exist
3275   }{
3276     \_stex_reset:N \l_stex_current_symbol_str
3277     \_stex_reset:N \arg
3278     \_stex_reset:N \comp
3279     \_stex_reset:N \l__stex_terms_custom_args_prop

```



```

3280     \bool_set_true:N \l_stex_allow_semantic_bool
3281   }
3282 }
3283
3284 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3285   \tl_if_empty:nTF {#2}{
3286     \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3287     \bool_set_true:N \l_tmpa_bool
3288     \bool_do_while:Nn \l_tmpa_bool {
3289       \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3290       \int_incr:N \l_tmpa_int
3291     }{
3292       \bool_set_false:N \l_tmpa_bool
3293     }
3294   }
3295   ){
3296     \int_set:Nn \l_tmpa_int { #2 }
3297   }
3298   \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3299   \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3300     \msg_error:nnxxx{stex}{error/overarity}
3301     {\int_use:N \l_tmpa_int}
3302     {\l_stex_current_symbol_str}
3303     {\str_count:N \l_tmpa_str}
3304   }
3305   \str_set:Nx \l_tmpa_str {\str_item:Nn \l_tmpa_str \l_tmpa_int}
3306   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3307     \bool_lazy_any:nF {
3308       {\str_if_eq_p:Vn \l_tmpa_str {a}}
3309       {\str_if_eq_p:Vn \l_tmpa_str {B}}
3310     }{
3311       \msg_error:nnxx{stex}{error/doubleargument}
3312       {\int_use:N \l_tmpa_int}
3313       {\l_stex_current_symbol_str}
3314     }
3315   }
3316   \exp_args:NNx \prop_put:Nnn \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {#3}
3317   \bool_set_true:N \l_stex_allow_semantic_bool
3318   \IfBooleanTF#1{
3319     \stex_annotate_invisible:n { %TODO
3320       \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3321     }
3322   }{ %TODO
3323     \exp_args:No \_stex_term_arg:nn {\l_tmpa_str\int_use:N \l_tmpa_int}{#3}
3324   }
3325   \bool_set_false:N \l_stex_allow_semantic_bool
3326 }
3327
3328
3329 \cs_new_protected:Nn \_stex_term_arg:nn {
3330   \bool_set_true:N \l_stex_allow_semantic_bool
3331   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3332   \bool_set_false:N \l_stex_allow_semantic_bool
3333 }

```

```

3334
3335 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3336   \exp_args:Nnx \use:nn
3337   { \int_set:Nn \l__stex_terms_downprec { #2 }
3338     \_stex_term_arg:nn { #1 }{ #3 }
3339   }
3340   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3341 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3342 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3343   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3344   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3345   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3346     \expandafter\if\expandafter\relax\noexpand#3
3347     \expandafter\_stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3
3348     \else\expandafter\_stex_terms_math_assoc_arg_simple:nn
3349     \expandafter{\expandafter}\expandafter#3\fi
3350   }{
3351     \_stex_terms_math_assoc_arg_simple:nn{#1}{#3}
3352   }
3353 }
3354
3355 \cs_new_protected:Nn \_stex_terms_math_assoc_arg_maybe_sequence:N {
3356   \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3357   \str_if_empty:NTF \l_tmpa_str {
3358     \exp_args:Nx \cs_if_eq:NNTF {
3359       \tl_head:N #1
3360     } \stex_invoke_sequence:n {
3361       \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3362       \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3363       \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq\_l_tmpa_str\_prop}{notation}}
3364       \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3365       \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3366         \exp_not:n{\exp_args:Nnx \use:nn} {
3367           \exp_not:n {
3368             \def\comp{\_varcomp}
3369             \str_set:Nn \l_stex_current_symbol_str
3370             } {varseq://\l_tmpa_str}
3371             \exp_not:n{ ##1 }
3372           }{
3373             \exp_not:n {
3374               \_stex_reset:N \comp
3375               \_stex_reset:N \l_stex_current_symbol_str
3376             }
3377           }
3378         }}}
3379       \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3380       \seq_reverse:N \l_tmpa_seq
3381       \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3382       \seq_map_inline:Nn \l_tmpa_seq {
3383         \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {

```

```

3384         \exp_args:Nno
3385         \l_tmpa_cs { ##1 } \l_tmpa_tl
3386     }
3387 }
3388 \tl_set:Nx \l_tmpa_tl {
3389     \stex_term_omv:nn {varseq://\l_tmpa_str}{
3390         \exp_args:No \exp_not:n \l_tmpa_tl
3391     }
3392 }
3393 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3394 }{
3395     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3396 }
3397 } {
3398     \stex_terms_math_assoc_arg_simple:nn{} { #1 }
3399 }
3400
3401 }
3402
3403 \cs_new_protected:Nn \stex_terms_math_assoc_arg_simple:nn {
3404     \clist_set:Nn \l_tmpa_clist{ #2 }
3405     \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3406         \tl_set:Nn \l_tmpa_tl { #2 }
3407     }{
3408         \clist_reverse:N \l_tmpa_clist
3409         \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3410         \tl_set:Nx \l_tmpa_tl { \stex_term_arg:nn{A#1}{
3411             \exp_args:No \exp_not:n \l_tmpa_tl
3412         }}
3413         \clist_map_inline:Nn \l_tmpa_clist {
3414             \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3415                 \exp_args:Nno
3416                 \l_tmpa_cs { \stex_term_arg:nn{A#1}{##1} } \l_tmpa_tl
3417             }
3418         }
3419     }
3420     \exp_args:No\l_tmpb_tl\l_tmpa_tl
3421 }

```

(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
3422 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3423 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3424 \int_new:N \l__stex_terms_downprec
3425 \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

```
3426 \tl_set:Nn \l__stex_terms_left_bracket_str (
3427 \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

```
3428 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3429   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3430     \bool_set_false:N \l__stex_terms_brackets_done_bool
3431     #2
3432   } {
3433     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3434       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3435         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3436         \dobrackets { #2 }
3437       }
3438     }{ #2 }
3439   }
3440 }
```

(End definition for __stex_terms_maybe_brackets:nn.)

\dobrackets

```
3441 \bool_new:N \l__stex_terms_brackets_done_bool
3442 %\RequirePackage{scalerel}
3443 \cs_new_protected:Npn \dobrackets #1 {
3444   %\ThisStyle{\if D\m@switch
3445   %   \exp_args:Nnx \use:nn
3446   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3447   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3448   % \else
3449   \exp_args:Nnx \use:nn
3450   {
3451     \bool_set_true:N \l__stex_terms_brackets_done_bool
3452     \int_set:Nn \l__stex_terms_downprec \infprec
3453     \l__stex_terms_left_bracket_str
3454     #1
3455   }
3456   {
3457     \bool_set_false:N \l__stex_terms_brackets_done_bool
3458     \l__stex_terms_right_bracket_str
3459     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3460   }
3461   %\fi}
3462 }
```

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

\withbrackets

```
3463 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3464   \exp_args:Nnx \use:nn
3465   {
3466     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
```

```

3467 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3468 #3
3469 }
3470 {
3471 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3472 {\l__stex_terms_left_bracket_str}
3473 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3474 {\l__stex_terms_right_bracket_str}
3475 }
3476 }

```

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

`\STEXinvisible`

```

3477 \cs_new_protected:Npn \STEXinvisible #1 {
3478 \stex_annotate_invisible:n { #1 }
3479 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

3480 \cs_new_protected:Nn \_stex_term_oms:nnn {
3481 \stex_annotate:nnn{ OMID }{ #2 }{
3482 \stex_highlight_term:nn { #1 } { #3 }
3483 }
3484 }
3485
3486 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3487 \_stex_terms_maybe_brackets:nn { #3 }{
3488 \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3489 }
3490 }

```

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

`_stex_term_math_omv:nn`

```

3491 \cs_new_protected:Nn \_stex_term_omv:nn {
3492 \stex_annotate:nnn{ OMV }{ #1 }{
3493 \stex_highlight_term:nn { #1 } { #2 }
3494 }
3495 }

```

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

`_stex_term_math_oma:nnnn`

```

3496 \cs_new_protected:Nn \_stex_term_oma:nnn {
3497 \stex_annotate:nnn{ OMA }{ #2 }{
3498 \stex_highlight_term:nn { #1 } { #3 }
3499 }
3500 }
3501
3502 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3503 \_stex_terms_maybe_brackets:nn { #3 }{
3504 \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }

```

```

3505 }
3506 }

```

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

`_stex_term_math_omb:nnnn`

```

3507 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3508   \stex_annotate:nnn{ OMBIND }{ #2 }{
3509     \stex_highlight_term:nn { #1 } { #3 }
3510   }
3511 }
3512
3513 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3514   \__stex_terms_maybe_brackets:nn { #3 }{
3515     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3516   }
3517 }

```

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

`\symref`
`\symname`

```

3518 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3519
3520 \keys_define:nn { stex / symname } {
3521   pre      .tl_set_x:N = \l__stex_terms_pre_tl ,
3522   post     .tl_set_x:N = \l__stex_terms_post_tl ,
3523   root     .tl_set_x:N = \l__stex_terms_root_tl
3524 }
3525
3526 \cs_new_protected:Nn \stex_symname_args:n {
3527   \tl_clear:N \l__stex_terms_post_tl
3528   \tl_clear:N \l__stex_terms_pre_tl
3529   \tl_clear:N \l__stex_terms_root_str
3530   \keys_set:nn { stex / symname } { #1 }
3531 }
3532
3533 \NewDocumentCommand \symref { m m }{
3534   \let\compemph_uri_prev:\compemph@uri
3535   \let\compemph@uri\symrefemph@uri
3536   \STEXsymbol{#1}!{ #2 }
3537   \let\compemph@uri\compemph_uri_prev:
3538 }
3539
3540 \NewDocumentCommand \synonym { 0{} m m }{
3541   \stex_symname_args:n { #1 }
3542   \let\compemph_uri_prev:\compemph@uri
3543   \let\compemph@uri\symrefemph@uri
3544   % TODO
3545   \STEXsymbol{#2}!{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3546   \let\compemph@uri\compemph_uri_prev:
3547 }
3548
3549 \NewDocumentCommand \symname { 0{} m }{
3550   \stex_symname_args:n { #1 }
3551   \stex_get_symbol:n { #2 }

```

```

3552 \str_set:Nx \l_tmpa_str {
3553   \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3554 }
3555 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3556
3557 \let\compemph_uri_prev:\compemph@uri
3558 \let\compemph@uri\symrefemph@uri
3559 \exp_args:NNx \use:nn
3560 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3561   \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3562 } }
3563 \let\compemph@uri\compemph_uri_prev:
3564 }
3565
3566 \NewDocumentCommand \Symname { 0{ } m }{
3567   \stex_symname_args:n { #1 }
3568   \stex_get_symbol:n { #2 }
3569   \str_set:Nx \l_tmpa_str {
3570     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3571   }
3572   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3573   \let\compemph_uri_prev:\compemph@uri
3574   \let\compemph@uri\symrefemph@uri
3575   \exp_args:NNx \use:nn
3576   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3577     \exp_after:wN \stex_capitalize:n \l_tmpa_str
3578     \l__stex_terms_post_tl
3579   } }
3580   \let\compemph@uri\compemph_uri_prev:
3581 }

```

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

30.3 Notation Components

```

3582 <@@=stex_notationcomps>
3583 \cs_new_protected:Nn \stex_highlight_term:nn {
3584   #2
3585 }
3586
3587 \cs_new_protected:Nn \stex_unhighlight_term:n {
3588   % \latexml_if:TF {
3589   %   #1
3590   % } {
3591   %   \rustex_if:TF {
3592   %     #1
3593   %   } {
3594     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3595   %   }
3596   % }
3597 }

```

`\stex_highlight_term:nn`

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

```

\comp
\compemph@uri 3598 \cs_new_protected:Npn \_comp #1 {
\compemph 3599 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 3600 \rustex_if:TF {
\defemph@uri 3601 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 3602 }{
\symrefemph@uri 3603 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
\varemp 3604 }
\varemp@uri 3605 }
3606 }
3607
3608 \cs_new_protected:Npn \_varcomp #1 {
3609 \str_if_empty:NF \l_stex_current_symbol_str {
3610 \rustex_if:TF {
3611 \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3612 }{
3613 \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }
3614 }
3615 }
3616 }
3617
3618 \def\comp{\_comp}
3619
3620 \cs_new_protected:Npn \compemph@uri #1 #2 {
3621 \compemph{ #1 }
3622 }
3623
3624
3625 \cs_new_protected:Npn \compemph #1 {
3626 #1
3627 }
3628
3629 \cs_new_protected:Npn \defemph@uri #1 #2 {
3630 \defemph{#1}
3631 }
3632
3633 \cs_new_protected:Npn \defemph #1 {
3634 \textbf{#1}
3635 }
3636
3637 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3638 \symrefemph{#1}
3639 }
3640
3641 \cs_new_protected:Npn \symrefemph #1 {
3642 \textbf{#1}
3643 }
3644
3645 \cs_new_protected:Npn \varemp@uri #1 #2 {
3646 \varemp{#1}
3647 }
3648

```



```

3649 \cs_new_protected:Npn \varemp #1 {
3650   #1
3651 }

```

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

`\ellipses`

```

3652 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
3653 \bool_new:N \l_stex_inarray_bool
3654 \bool_set_false:N \l_stex_inarray_bool
3655 \NewDocumentCommand \parray { m m } {
3656   \begingroup
3657   \bool_set_true:N \l_stex_inarray_bool
3658   \begin{array}{#1}
3659     #2
3660   \end{array}
3661   \endgroup
3662 }
3663
3664 \NewDocumentCommand \prmatrix { m } {
3665   \begingroup
3666   \bool_set_true:N \l_stex_inarray_bool
3667   \begin{matrix}
3668     #1
3669   \end{matrix}
3670   \endgroup
3671 }
3672
3673 \def \maybepline {
3674   \bool_if:NT \l_stex_inarray_bool {\hline}
3675 }
3676
3677 \def \parrayline #1 #2 {
3678   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3679 }
3680
3681 \def \pmrow #1 { \parrayline{ }{ #1 } }
3682
3683 \def \parraylineh #1 #2 {
3684   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3685 }
3686
3687 \def \parraycell #1 {
3688   #1 \bool_if:NT \l_stex_inarray_bool {&}
3689 }

```

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

30.4 Variables

3690 <@@=stex_variables>

\stex_invoke_variable:n Invokes a variable

```

3691 \cs_new_protected:Nn \stex_invoke_variable:n {
3692   \if_mode_math:
3693     \exp_after:wN \__stex_variables_invoke_math:n
3694   \else:
3695     \exp_after:wN \__stex_variables_invoke_text:n
3696   \fi: {#1}
3697 }
3698
3699 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3700   %TODO
3701 }
3702
3703
3704 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3705   \peek_charcode_remove:NTF ! {
3706     \peek_charcode_remove:NTF ! {
3707       \peek_charcode:NTF [ {
3708         \__stex_variables_invoke_op_custom:nw
3709       }{
3710         % TODO throw error
3711       }
3712     }{
3713       \__stex_variables_invoke_op:n { #1 }
3714     }
3715   }{
3716     \peek_charcode_remove:NTF * {
3717       \__stex_variables_invoke_text:n { #1 }
3718     }{
3719       \__stex_variables_invoke_math_ii:n { #1 }
3720     }
3721   }
3722 }
3723
3724 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3725   \cs_if_exist:cTF {
3726     stex_var_op_notation_ #1 _cs
3727   }{
3728     \exp_args:Nnx \use:nn {
3729       \def\comp{\_varcomp}
3730       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3731       \stex_term_omv:nn { var://#1 }{
3732         \use:c{stex_var_op_notation_ #1 _cs }
3733       }
3734     }{
3735       \stex_reset:N \comp
3736       \stex_reset:N \l_stex_current_symbol_str
3737     }
3738   }{
3739     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{

```

```

3740     \stex_variables_invoke_math_ii:n {#1}
3741   }{
3742     \msg_error:nnxx{stex}{error/noop}{variable~#1}{ }
3743   }
3744 }
3745 }
3746
3747 \cs_new_protected:Npn \stex_variables_invoke_math_ii:n #1 {
3748   \cs_if_exist:cTF {
3749     stex_var_notation_#1_cs
3750   }{
3751     \tl_set:Nx \stex_symbol_after_invokation_tl {
3752       \stex_reset:N \comp
3753       \stex_reset:N \stex_symbol_after_invokation_tl
3754       \stex_reset:N \l_stex_current_symbol_str
3755       \bool_set_true:N \l_stex_allow_semantic_bool
3756     }
3757     \def\comp{\_varcomp}
3758     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3759     \bool_set_false:N \l_stex_allow_semantic_bool
3760     \use:c{stex_var_notation_#1_cs}
3761   }{
3762     \msg_error:nnxx{stex}{error/nonotation}{variable~#1}{s}
3763   }
3764 }

```

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

30.5 Sequences

```

3765 <@@=stex_sequences>
3766
3767 \cs_new_protected:Nn \stex_invoke_sequence:n {
3768   \peek_charcode_remove:NTF ! {
3769     \stex_term_omv:nn {varseq://#1}{
3770       \exp_args:Nnx \use:nn {
3771         \def\comp{\_varcomp}
3772         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3773         \prop_item:cn{stex_varseq_#1_prop}{notation}
3774       }{
3775         \stex_reset:N \comp
3776         \stex_reset:N \l_stex_current_symbol_str
3777       }
3778     }
3779   }{
3780     \bool_set_false:N \l_stex_allow_semantic_bool
3781     \def\comp{\_varcomp}
3782     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3783     \tl_set:Nx \stex_symbol_after_invokation_tl {
3784       \stex_reset:N \comp
3785       \stex_reset:N \stex_symbol_after_invokation_tl
3786       \stex_reset:N \l_stex_current_symbol_str
3787       \bool_set_true:N \l_stex_allow_semantic_bool
3788     }

```

```
3789     \use:c { stex_varseq_#1_cs }
3790   }
3791 }
3792 </package>
```

Chapter 31

STEX -Structural Features Implementation

```
3793 ⟨*package⟩
3794
3795 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3796
    Warnings and error messages
3797 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3798   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3799 }
3800 \msg_new:nnn{stex}{error/interpretmodule/nodfiniens}{
3801   Symbol~#1~not~assigned~in~interpretmodule~#2
3802 }
3803
3804 \msg_new:nnn{stex}{error/unknownstructure}{
3805   No~structure~#1~found!
3806 }
3807
3808 \msg_new:nnn{stex}{error/unknownfield}{
3809   No~field~#1~in~instance~#2~found!\#3
3810 }
3811
3812 \msg_new:nnn{stex}{error/keyval}{
3813   Invalid~key=value~pair~#1
3814 }
3815 \msg_new:nnn{stex}{error/instantiate/missing}{
3816   Assignments~missing~in~instantiate:~#1
3817 }
3818 \msg_new:nnn{stex}{error/incompatible}{
3819   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3820 }
3821
```

31.1 Imports with modification

```

3822 <@@=stex_copymodule>
3823 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3824   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3825     \tl_set:Nn \l_tmpa_tl { #1 }
3826     \__stex_copymodule_get_symbol_from_cs:
3827   }{
3828     % argument is a string
3829     % is it a command name?
3830     \cs_if_exist:cTF { #1 }{
3831       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3832       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3833       \str_if_empty:NNTF \l_tmpa_str {
3834         \exp_args:Nx \cs_if_eq:NNTF {
3835           \tl_head:N \l_tmpa_tl
3836         } \stex_invoke_symbol:n {
3837           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3838         }{
3839           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3840         }
3841       } {
3842         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3843       }
3844     }{
3845       % argument is not a command name
3846       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3847       % \l_stex_all_symbols_seq
3848     }
3849   }
3850 }
3851
3852 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3853   \str_set:Nn \l_tmpa_str { #1 }
3854   \bool_set_false:N \l_tmpa_bool
3855   \bool_if:NF \l_tmpa_bool {
3856     \tl_set:Nn \l_tmpa_tl {
3857       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3858     }
3859     \str_set:Nn \l_tmpa_str { #1 }
3860     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3861     \seq_map_inline:Nn #2 {
3862       \str_set:Nn \l_tmpb_str { ##1 }
3863       \str_if_eq:eeT { \l_tmpa_str } {
3864         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3865       } {
3866         \seq_map_break:n {
3867           \tl_set:Nn \l_tmpa_tl {
3868             \str_set:Nn \l_stex_get_symbol_uri_str {
3869               ##1
3870             }
3871           }
3872         }
3873       }

```

```

3874     }
3875     \l_tmpa_tl
3876   }
3877 }
3878
3879 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3880   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3881     { \tl_tail:N \l_tmpa_tl }
3882   \tl_if_single:NTF \l_tmpa_tl {
3883     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3884       \exp_after:wN \str_set:Nn \exp_after:wN
3885         \l_stex_get_symbol_uri_str \l_tmpa_tl
3886       \__stex_copymodule_get_symbol_check:n { #1 }
3887     }{
3888       % TODO
3889       % tail is not a single group
3890     }
3891   }{
3892     % TODO
3893     % tail is not a single group
3894   }
3895 }
3896
3897 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3898   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3899     \msg_error:nnxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3900       :~\seq_use:Nn #1 {,~}
3901     }
3902   }
3903 }
3904
3905 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3906   \stex_import_module_uri:nn { #1 } { #2 }
3907   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3908   \stex_import_require_module:nnnn
3909     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3910     { \l_stex_import_path_str } { \l_stex_import_name_str }
3911   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3912   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3913   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3914   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3915     \seq_map_inline:cn {c_stex_module_###_constants}{
3916       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3917         ##1 ? ####1
3918       }
3919     }
3920   }
3921   \seq_clear:N \l_tmpa_seq
3922   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3923     name      = \l_stex_current_copymodule_name_str ,
3924     module    = \l_stex_current_module_str ,
3925     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3926     includes  = \l_tmpa_seq ,
3927     fields    = \l_tmpa_seq

```

```

3928 }
3929 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3930 as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3931 \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3932 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3933 \stex_if_smsmode:F {
3934 \begin{stex_annotate_env} {#4} {
3935 \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3936 }
3937 \stex_annotate_invisible:nnn{domain}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}}
3938 }
3939 %\bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3940 %\bool_set_false:N \_stex_html_do_output_bool
3941 }
3942 \cs_new_protected:Nn \stex_copymodule_end:n {
3943 \def \l_tmpa_cs ##1 ##2 {#1}
3944 %\bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3945 \tl_clear:N \l_tmpa_tl
3946 \tl_clear:N \l_tmpb_tl
3947 \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3948 \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3949 \seq_map_inline:cn {c_stex_module_##1_constants}{
3950 \tl_clear:N \l_tmpc_tl
3951 \l_tmpa_cs{##1}{####1}
3952 \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3953 \tl_put_right:Nx \l_tmpa_tl {
3954 \prop_set_from_keyval:cn {
3955 l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3956 }{
3957 \exp_after:wN \prop_to_keyval:N \csname
3958 l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3959 \endcsname
3960 }
3961 \seq_clear:c {
3962 l_stex_symdecl_
3963 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3964 _notations
3965 }
3966 }
3967 \tl_put_right:Nx \l_tmpc_tl {
3968 \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3969 \stex_if_smsmode:F{\stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
3970 }
3971 \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3972 \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3973 \tl_put_right:Nx \l_tmpc_tl {
3974 \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}}
3975 }
3976 \tl_put_right:Nx \l_tmpa_tl {
3977 \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3978 \stex_invoke_symbol:n {
3979 \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3980 }
3981 }

```



```

3982     }
3983   }
3984 }{
3985   \tl_put_right:Nx \l_tmpc_tl {
3986     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3987   }
3988   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3989   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3990   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3991   \tl_put_right:Nx \l_tmpa_tl {
3992     \prop_set_from_keyval:cn {
3993       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3994     }{
3995       \prop_to_keyval:N \l_tmpa_prop
3996     }
3997     \seq_clear:c {
3998       l_stex_symdecl_
3999       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
4000       _notations
4001     }
4002   }
4003   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
4004   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
4005     \tl_put_right:Nx \l_tmpc_tl {
4006       \stex_if_smsmode:F{\stex_annotate_invisible:nnn{macroname}}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4007     }
4008     \tl_put_right:Nx \l_tmpa_tl {
4009       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
4010         \stex_invoke_symbol:n {
4011           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
4012         }
4013       }
4014     }
4015   }
4016 }
4017 \stex_if_smsmode:F{
4018   \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
4019     \tl_put_right:Nx \l_tmpc_tl {
4020       $\stex_annotate_invisible:nnn{definiens}}{\exp_after:wN \exp_not:N\csname l__stex_copymodule_copymodule_##1?####1_def_tl\endcsname}
4021     }
4022   }
4023   \tl_put_right:Nx \l_tmpb_tl {
4024     \stex_annotate:nnn{assignment} {##1?####1} { \exp_after:wN \exp_not:n \exp_after:wN }
4025   }
4026 }
4027 }
4028 }
4029 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
4030 \tl_put_left:Nx \l_tmpa_tl {
4031   \prop_set_from_keyval:cn {
4032     l_stex_copymodule_ \l_stex_current_module_str?\l_stex_current_copymodule_name_str _prop
4033   }{
4034     \prop_to_keyval:N \l_stex_current_copymodule_prop
4035   }

```

```

4036 }
4037 \seq_gput_right:cx{c_stex_module_\l_stex_current_module_str _copymodules}{
4038   \l_stex_current_module_str?\l_stex_current_copymodule_name_str
4039 }
4040 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl
4041 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
4042 \exp_args:Nx \stex_do_up_to_module:n {
4043   \exp_args:No \exp_not:n \l_tmpa_tl
4044 }
4045 \stex_debug:nn{copymodule}{output:\meaning \l_tmpb_tl}
4046 \l_tmpb_tl
4047 \stex_if_smsmode:F {
4048   \end{stex_annotate_env}
4049 }
4050 }
4051
4052 \NewDocumentEnvironment {copymodule} { 0{} m m}{
4053   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ copymodule }
4054   \stex_deactivate_macro:Nn \symdecl {module~environments}
4055   \stex_deactivate_macro:Nn \symdef {module~environments}
4056   \stex_deactivate_macro:Nn \notation {module~environments}
4057   \stex_reactivate_macro:N \assign
4058   \stex_reactivate_macro:N \renamedekl
4059   \stex_reactivate_macro:N \donotcopy
4060   \stex_smsmode_do:
4061 }{
4062   \stex_copymodule_end:n {}
4063 }
4064
4065 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
4066   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ interpretmodule }
4067   \stex_deactivate_macro:Nn \symdecl {module~environments}
4068   \stex_deactivate_macro:Nn \symdef {module~environments}
4069   \stex_deactivate_macro:Nn \notation {module~environments}
4070   \stex_reactivate_macro:N \assign
4071   \stex_reactivate_macro:N \renamedekl
4072   \stex_reactivate_macro:N \donotcopy
4073   \stex_smsmode_do:
4074 }{
4075   \stex_copymodule_end:n {
4076     \tl_if_exist:cF {
4077       l__stex_copymodule_copymodule_##1?##2_def_tl
4078     }{
4079       \str_if_eq:eeF {
4080         \prop_item:cn{
4081           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4082         }{ true }{
4083           \msg_error:nnxx{stex}{error/interpretmodule/nodéfiniens}{
4084             ##1?##2
4085           }{\l_stex_current_copymodule_name_str}
4086         }
4087       }
4088     }
4089   }

```

```

4090
4091 \NewDocumentCommand \donotcopy { m }{
4092   \str_clear:N \l_stex_import_name_str
4093   \str_set:Nn \l_tmpa_str { #1 }
4094   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4095   \seq_map_inline:Nn \l_stex_all_modules_seq {
4096     \str_set:Nn \l_tmpb_str { ##1 }
4097     \str_if_eq:eeT { \l_tmpa_str } {
4098       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4099     } {
4100       \seq_map_break:n {
4101         \stex_if_do_html:T {
4102           \stex_if_smsmode:F {
4103             \stex_annotate_invisible:nnn{donotcopy}{##1}{
4104               \stex_annotate:nnn{domain}{##1}{}}
4105           }
4106         }
4107       }
4108       \str_set_eq:NN \l_stex_import_name_str \l_tmpb_str
4109     }
4110   }
4111   \seq_map_inline:cn {c_stex_module_###1_copymodules}{
4112     \str_set:Nn \l_tmpb_str { #####1 }
4113     \str_if_eq:eeT { \l_tmpa_str } {
4114       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
4115     } {
4116       \seq_map_break:n {\seq_map_break:n {
4117         \stex_if_do_html:T {
4118           \stex_if_smsmode:F {
4119             \stex_annotate_invisible:nnn{donotcopy}{#####1}{
4120               \stex_annotate:nnn{domain}{
4121                 \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4122               }{}}
4123             }
4124           }
4125         }
4126         \str_set:Nx \l_stex_import_name_str {
4127           \prop_item:cn {l_stex_copymodule_ #####1 _prop}{module}
4128         }
4129       }}
4130     }
4131   }
4132 }
4133 \str_if_empty:NTF \l_stex_import_name_str {
4134   % TODO throw error
4135 }{
4136   \stex_collect_imports:n {\l_stex_import_name_str }
4137   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4138     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }
4139     \seq_map_inline:cn {c_stex_module_###1_constants}{
4140       \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4141       \bool_lazy_any:nT {
4142         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_name_str}}
4143         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?#####1_macroname_str}}

```

```

4144         { \cs_if_exist_p:c {l__stex_copymodule_copymodule_###1?####1_def_tl}}
4145     }{
4146         % TODO throw error
4147     }
4148 }
4149 }
4150 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4151 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_name_str }
4152 \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4153 }
4154 \stex_smsmode_do:
4155 }
4156
4157 \NewDocumentCommand \assign { m m }{
4158     \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4159     \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4160     \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4161     \stex_smsmode_do:
4162 }
4163
4164 \keys_define:nn { stex / renamedecl } {
4165     name .str_set_x:N = \l_stex_renamedecl_name_str
4166 }
4167 \cs_new_protected:Nn \__stex_copymodule_renamedecl_args:n {
4168     \str_clear:N \l_stex_renamedecl_name_str
4169     \keys_set:nn { stex / renamedecl } { #1 }
4170 }
4171
4172 \NewDocumentCommand \renamedecl { 0{} m m }{
4173     \__stex_copymodule_renamedecl_args:n { #1 }
4174     \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4175     \stex_debug:nn{renamedecl}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4176     \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4177     \str_if_empty:NTF \l_stex_renamedecl_name_str {
4178         \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4179             \l_stex_get_symbol_uri_str
4180         } }
4181     } {
4182         \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4183             \stex_debug:nn{renamedecl}{@~\l_stex_current_module_str ? \l_stex_renamedecl_name_str}
4184             \prop_set_eq:cc {l_stex_symdecl_
4185                 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4186                 _prop
4187             }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}
4188             \seq_set_eq:cc {l_stex_symdecl_
4189                 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4190                 _notations
4191             }{l_stex_symdecl_ \l_stex_get_symbol_uri_str _notations}
4192             \prop_put:cnx {l_stex_symdecl_
4193                 \l_stex_current_module_str ? \l_stex_renamedecl_name_str
4194                 _prop
4195             }{ name }{ \l_stex_renamedecl_name_str }
4196             \prop_put:cnx {l_stex_symdecl_
4197                 \l_stex_current_module_str ? \l_stex_renamedecl_name_str

```

```

4198     _prop
4199   }{ module }{ \l_stex_current_module_str }
4200   \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4201     \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4202   }
4203   \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4204     \l_stex_current_module_str ? \l_stex_renameddecl_name_str
4205   } }
4206 }
4207 \stex_smsmode_do:
4208 }
4209
4210 \stex_deactivate_macro:Nn \assign {copymodules}
4211 \stex_deactivate_macro:Nn \renameddecl {copymodules}
4212 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4213
4214
4215 \seq_new:N \l_stex_implicit_morphisms_seq
4216 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4217   \stex_import_module_uri:nn { #1 } { #2 }
4218   \stex_debug:nn{implicits}{
4219     Implicit~morphism:~
4220     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4221   }
4222   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4223     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4224   }{
4225     \msg_error:nnn{stex}{error/conflictingmodules}{
4226       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4227     }
4228   }
4229 }
4230 % TODO
4231
4232
4233
4234 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4235   \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4236 }
4237 }
4238

```

31.2 The feature environment

structural@feature

```

4239 <@@=stex_features>
4240
4241 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4242   \stex_if_in_module:F {
4243     \msg_set:nnn{stex}{error/nomodule}{
4244       Structural~Feature~has~to~occur~in~a~module:\\
4245       Feature~#2~of~type~#1\\
4246       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq

```

```

4247   }
4248   \msg_error:nn{stex}{error/nomodule}
4249 }
4250
4251 \str_set_eq:NN \l_tmpa_str \l_stex_current_module_str
4252
4253 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4254
4255 \stex_if_smsmode:F {
4256   \begin{stex_annotate_env}{ feature:#1 }{\l_tmpa_str ? #2 - #1}
4257   \stex_annotate_invisible:nnn{header}{}{ #3 }
4258 }
4259 }{
4260   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4261   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4262   \stex_debug:nn{features}{
4263     Feature: \l_stex_last_feature_str
4264   }
4265   \stex_if_smsmode:F {
4266     \end{stex_annotate_env}
4267   }
4268 }

```

31.3 Structure

structure

```

4269 <@@=stex_structures>
4270 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4271   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4272     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4273   }
4274   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4275     {#1}{#2}
4276 }
4277
4278 \keys_define:nn { stex / features / structure } {
4279   name .str_set_x:N = \l__stex_structures_name_str ,
4280 }
4281
4282 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4283   \str_clear:N \l__stex_structures_name_str
4284   \keys_set:nn { stex / features / structure } { #1 }
4285 }
4286
4287 \NewDocumentEnvironment{mathstructure}{m O{}}{
4288   \__stex_structures_structure_args:n { #2 }
4289   \str_if_empty:NT \l__stex_structures_name_str {
4290     \str_set:Nx \l__stex_structures_name_str { #1 }
4291   }
4292   \stex_suppress_html:n {
4293     \exp_args:Nx \stex_symdecl_do:nn {
4294       name = \l__stex_structures_name_str ,
4295       def = {\STEXsymbol{module-type}}{

```

```

4296     \stex_term_math_oms:nnnn {
4297         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4298         { ns } ?
4299         \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4300         { name } / \l__stex_structures_name_str - structure
4301     }{}{}{}
4302 }}
4303 }{ #1 }
4304 }
4305 \exp_args:Nnnx
4306 \begin{structural_feature_module}{ structure }
4307 { \l__stex_structures_name_str }{}
4308 \stex_smsmode_do:
4309 }{
4310     \end{structural_feature_module}
4311     \stex_reset_up_to_module:
4312     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4313     \seq_clear:N \l_tmpa_seq
4314     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4315         \seq_map_inline:cn{c_stex_module_##1_constants}{
4316             \seq_put_right:Nn \l_tmpa_seq { ##1 ? #####1 }
4317         }
4318     }
4319     \exp_args:Nnno
4320     \prop_gput:cn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4321     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4322     \stex_add_structure_to_current_module:nn
4323         \l__stex_structures_name_str
4324         \l_stex_last_feature_str
4325     \exp_args:Nx
4326     \stex_add_to_current_module:n {
4327         \tl_set:cn { #1 }{
4328             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4329         }
4330     }
4331     \exp_args:Nx
4332     \stex_do_up_to_module:n {
4333         \tl_set:cn { #1 }{
4334             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures_name_str }
4335         }
4336     }
4337 }
4338
4339 \cs_new:Nn \stex_invoke_structure:nn {
4340     \stex_invoke_symbol:n { #1?#2 }
4341 }
4342
4343 \cs_new_protected:Nn \stex_get_structure:n {
4344     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4345         \tl_set:Nn \l_tmpa_tl { #1 }
4346         \__stex_structures_get_from_cs:
4347     }{
4348         \cs_if_exist:cTF { #1 }{
4349             \cs_set_eq:Nc \l_tmpa_cs { #1 }

```

```

4350 \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4351 \str_if_empty:NTF \l_tmpa_str {
4352   \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4353     \__stex_structures_get_from_cs:
4354   }{
4355     \__stex_structures_get_from_string:n { #1 }
4356   }
4357 }{
4358   \__stex_structures_get_from_string:n { #1 }
4359 }
4360 }{
4361   \__stex_structures_get_from_string:n { #1 }
4362 }
4363 }
4364 }
4365
4366 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4367   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4368     { \tl_tail:N \l_tmpa_tl }
4369   \str_set:Nx \l_tmpa_str {
4370     \exp_after:wN \use_i:nn \l_tmpa_tl
4371   }
4372   \str_set:Nx \l_tmpb_str {
4373     \exp_after:wN \use_ii:nn \l_tmpa_tl
4374   }
4375   \str_set:Nx \l_stex_get_structure_str {
4376     \l_tmpa_str ? \l_tmpb_str
4377   }
4378   \str_set:Nx \l_stex_get_structure_module_str {
4379     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4380   }
4381 }
4382
4383 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4384   \tl_set:Nn \l_tmpa_tl {
4385     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4386   }
4387   \str_set:Nn \l_tmpa_str { #1 }
4388   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4389
4390   \seq_map_inline:Nn \l_stex_all_modules_seq {
4391     \prop_if_exist:cT {c_stex_module_##1_structures} {
4392       \prop_map_inline:cn {c_stex_module_##1_structures} {
4393         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4394           \prop_map_break:n{\seq_map_break:n{
4395             \tl_set:Nn \l_tmpa_tl {
4396               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4397               \str_set:Nn \l_stex_get_structure_module_str {####2}
4398             }
4399           }}
4400         }
4401       }
4402     }
4403   }

```



```

4404 \l_tmpa_tl
4405 }

\instantiate

4406
4407 \keys_define:nn { stex / instantiate } {
4408   name .str_set_x:N = \l__stex_structures_name_str
4409 }
4410 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4411   \str_clear:N \l__stex_structures_name_str
4412   \keys_set:nn { stex / instantiate } { #1 }
4413 }
4414
4415 \NewDocumentCommand \instantiate {m O{} m m m}{
4416   \begin_group
4417     \stex_get_structure:n {#4}
4418     \__stex_structures_instantiate_args:n { #2 }
4419     \str_if_empty:NT \l__stex_structures_name_str {
4420       \str_set:Nn \l__stex_structures_name_str { #1 }
4421     }
4422     \seq_clear:N \l__stex_structures_fields_seq
4423     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4424     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4425       \seq_map_inline:cn {c_stex_module_##1_constants}{
4426         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4427       }
4428     }
4429     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4430     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4431     \prop_clear:N \l_tmpa_prop
4432     \seq_map_inline:Nn \l_tmpa_seq {
4433       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4434       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4435         \msg_error:nnn{stex}{error/keyval}{##1}
4436       }
4437       \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4438       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4439       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4440       \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4441       \exp_args:Nxx \str_if_eq:nnF
4442         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4443         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4444         \msg_error:nnxxx{stex}{error/incompatible}
4445         {\l__stex_structures_dom_str
4446         {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4447         {\l_stex_get_symbol_uri_str
4448         {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4449       }
4450       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4451     }
4452     \seq_if_empty:NF \l__stex_structures_fields_seq {
4453       \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn\l__stex_structures_fields_
4454     }
4455     \exp_args:Nx

```

```

4456 \stex_add_to_current_module:n {
4457   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4458   domain = \l_stex_get_structure_module_str ,
4459   \prop_to_keyval:N \l_tmpa_prop
4460 }
4461 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4462 }
4463 \exp_args:Nx
4464 \stex_do_up_to_module:n {
4465   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4466   domain = \l_stex_get_structure_module_str ,
4467   \prop_to_keyval:N \l_tmpa_prop
4468 }
4469 \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4470 }
4471 \stex_debug:nn{instantiate}{
4472   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4473   \prop_to_keyval:N \l_tmpa_prop
4474 }
4475 \exp_args:Nxx \stex_symdecl_do:nn {
4476   type={\STEXsymbol{module-type}}{
4477     \stex_term_math_oms:nnnn {
4478       \l_stex_get_structure_module_str
4479     }{}{0}{}
4480   }}
4481   {}{\l__stex_structures_name_str}
4482   \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4483 \endgroup
4484 \stex_smsmode_do:\ignorespacesandpars
4485 }
4486
4487 \cs_new_protected:Nn \stex_symbol_or_var:n {
4488   \cs_if_exist:CTF{#1}{
4489     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4490     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4491     \str_if_empty:NTF \l_tmpa_str {
4492       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4493       \stex_invoke_variable:n {
4494         \bool_set_true:N \l_stex_symbol_or_var_bool
4495         \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4496         \str_set:Nx \l_stex_get_symbol_uri_str {
4497           \exp_after:wN \use:n \l_tmpa_tl
4498         }
4499       }{
4500         \bool_set_false:N \l_stex_symbol_or_var_bool
4501         \stex_get_symbol:n{#1}
4502       }
4503     }{
4504       \__stex_structures_symbolorvar_from_string:n{ #1 }
4505     }
4506   }{
4507     \__stex_structures_symbolorvar_from_string:n{ #1 }
4508   }
4509 }

```

```

4510 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4511   \prop_if_exist:cTF {l_stex_variable_#1_prop}{
4512     \bool_set_true:N \l_stex_symbol_or_var_bool
4513     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4514   }{
4515     \bool_set_false:N \l_stex_symbol_or_var_bool
4516     \stex_get_symbol:n{#1}
4517   }
4518 }
4519 }
4520
4521 \keys_define:nn { stex / varinstantiate } {
4522   name      .str_set_x:N = \l__stex_structures_name_str,
4523   bind      .choices:nn =
4524     {forall,exists}
4525     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4526 }
4527 }
4528 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4529   \str_clear:N \l__stex_structures_name_str
4530   \str_clear:N \l__stex_structures_bind_str
4531   \keys_set:nn { stex / varinstantiate } { #1 }
4532 }
4533
4534 \NewDocumentCommand \varinstantiate {m O{}} m m m m){
4535   \beginingroup
4536     \stex_get_structure:n {#4}
4537     \__stex_structures_varinstantiate_args:n { #2 }
4538     \str_if_empty:NT \l__stex_structures_name_str {
4539       \str_set:Nn \l__stex_structures_name_str { #1 }
4540     }
4541     \stex_if_do_html:TF{
4542       \stex_annotate:nnn{varinstance}{\l__stex_structures_name_str}
4543     }{\use:n}
4544     {
4545       \stex_if_do_html:T{
4546         \stex_annotate:nnn{domain}{\l_stex_get_structure_module_str}{ }
4547       }
4548       \seq_clear:N \l__stex_structures_fields_seq
4549       \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4550       \seq_map_inline:Nn \l_stex_collect_imports_seq {
4551         \seq_map_inline:cn {c_stex_module_##1_constants}{
4552           \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? ####1 }
4553         }
4554       }
4555       \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4556       \prop_clear:N \l_tmpa_prop
4557       \tl_if_empty:nF {#3} {
4558         \seq_set_split:Nnn \l_tmpa_seq , {#3}
4559         \seq_map_inline:Nn \l_tmpa_seq {
4560           \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4561           \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4562             \msg_error:nnn{stex}{error/keyval}{##1}
4563           }

```

```

4564 \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structures_dom_str
4565 \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4566 \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri_str
4567 \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4568 \stex_if_do_html:T{
4569   \stex_annotate:nnn{assign}{\l__stex_structures_dom_str,\l_stex_get_symbol_uri_str}
4570 }
4571 \bool_if:NTF \l_stex_symbol_or_var_bool {
4572   \exp_args:Nxx \str_if_eq:nnF
4573     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4574     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}{
4575     \msg_error:nnxxx{stex}{error/incompatible}
4576     {\l__stex_structures_dom_str}
4577     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4578     {\l_stex_get_symbol_uri_str}
4579     {\prop_item:cn{l_stex_variable\l_stex_get_symbol_uri_str _prop}{args}}
4580   }
4581   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4582 }}{
4583   \exp_args:Nxx \str_if_eq:nnF
4584     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4585     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}{
4586     \msg_error:nnxxx{stex}{error/incompatible}
4587     {\l__stex_structures_dom_str}
4588     {\prop_item:cn{l_stex_symdecl\l__stex_structures_dom_str _prop}{args}}
4589     {\l_stex_get_symbol_uri_str}
4590     {\prop_item:cn{l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{args}}
4591   }
4592   \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {
4593 }}{
4594 }
4595 }
4596 \tl_gclear:N \g__stex_structures_aftergroup_tl
4597 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4598   \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl\l__stex_structures_fields_seq #1} \l_stex_get_symbol_uri_str}
4599   \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4600   \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4601     \stex_find_notation:nn{##1}{}
4602     \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4603       {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4604     \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tmpa_str _cs}}
4605     \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4606       \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4607       {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4608       \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structures_tmpa_op_\l_tmpa_str _cs}}
4609     }
4610   }
4611 }
4612 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4613   \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4614     name    = \l_tmpa_str ,
4615     args    = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4616     arity   = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4617     assocs  = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}

```

```

4618     }
4619     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str_cs}
4620     {g__stex_structures_tmpa_\l_tmpa_str_cs}
4621     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str_cs}
4622     {g__stex_structures_tmpa_op_\l_tmpa_str_cs}
4623   }
4624   \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_inv
4625 }
4626 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4627   \prop_set_from_keyval:cn {l_stex_varinstance_\l__stex_structures_name_str_prop }{
4628     domain = \l_stex_get_structure_module_str ,
4629     \prop_to_keyval:N \l_tmpa_prop
4630   }
4631   \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4632   \tl_set:cn {l_stex_varinstance_\l__stex_structures_name_str_op_tl}{
4633     \exp_args:Nnx \exp_not:N \use:nn {
4634       \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_nam
4635       \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4636         \exp_not:n{
4637           \_varcomp{#5}
4638         }
4639       }
4640     }{
4641       \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4642     }
4643   }
4644 }
4645 }
4646 \stex_debug:nn{varinstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_a
4647 \aftergroup\g__stex_structures_aftergroup_tl
4648 \endgroup
4649 \stex_smsmode_do:\ignorespacesandpars
4650 }
4651
4652 \cs_new_protected:Nn \stex_invoke_instance:n {
4653   \peek_charcode_remove:NTF ! {
4654     \stex_invoke_symbol:n{#1}
4655   }{
4656     \_stex_invoke_instance:nn {#1}
4657   }
4658 }
4659
4660
4661 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4662   \peek_charcode_remove:NTF ! {
4663     \exp_args:Nnx \use:nn {
4664       \def\comp{\_varcomp}
4665       \use:c{l_stex_varinstance_#1_op_tl}
4666     }{
4667       \_stex_reset:N \comp
4668     }
4669   }{
4670     \_stex_invoke_varinstance:nn {#1}
4671   }

```

```

4672 }
4673
4674 \cs_new_protected:Nn \stex_invoke_instance:nn {
4675   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4676     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4677   }{
4678     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4679     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
4680       \prop_to_keyval:N \l_tmpa_prop
4681     }
4682   }
4683 }
4684
4685 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4686   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4687     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4688     \l_tmpa_tl
4689   }{
4690     \msg_error:nnnnn{stex}{error/unknownfield}{#2}{#1}{
4691     }
4692   }

```

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

\stex_invoke_structure:nnn

```

4693 % #1: URI of the instance
4694 % #2: URI of the instantiated module
4695 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4696   \tl_if_empty:nTF{ #3 }{
4697     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4698       c_stex_feature_ #2 _prop
4699     }
4700     \tl_clear:N \l_tmpa_tl
4701     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4702     \seq_map_inline:Nn \l_tmpa_seq {
4703       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4704       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4705       \cs_if_exist:cT {
4706         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4707       }{
4708         \tl_if_empty:NF \l_tmpa_tl {
4709           \tl_put_right:Nn \l_tmpa_tl {,}
4710         }
4711         \tl_put_right:Nx \l_tmpa_tl {
4712           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4713         }
4714       }
4715     }
4716     \exp_args:No \mathstruct \l_tmpa_tl
4717   }{
4718     \stex_invoke_symbol:n{#1/#3}
4719   }
4720 }

```

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

4721 `\endpackage`

Chapter 32

STEX -Statements Implementation

```
4722 <*package>
4723
4724 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4725
4726 <@@=stex_statements>
    Warnings and error messages
4727

\titleemph
4728 \def\titleemph#1{\textbf{#1}}

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

32.1 Definitions

definiendum

```
4729 \keys_define:nn {stex / definiendum }{
4730   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4731   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4732   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4733   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4734 }
4735 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4736   \str_clear:N \l__stex_statements_definiendum_root_str
4737   \tl_clear:N \l__stex_statements_definiendum_post_tl
4738   \str_clear:N \l__stex_statements_definiendum_gfa_str
4739   \keys_set:nn { stex / definiendum }{ #1 }
4740 }
4741 \NewDocumentCommand \definiendum { O{} m m } {
4742   \__stex_statements_definiendum_args:n { #1 }
4743   \stex_get_symbol:n { #2 }
4744   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4745   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4746     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```



```

4747     \tl_set:Nn \l_tmpa_tl { #3 }
4748   } {
4749     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4750     \tl_set:Nn \l_tmpa_tl {
4751       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4752     }
4753   }
4754 } {
4755   \tl_set:Nn \l_tmpa_tl { #3 }
4756 }
4757
4758 % TODO root
4759 \rustex_if:TF {
4760   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4761 } {
4762   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4763 }
4764 }
4765 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4766
4767 \NewDocumentCommand \definame { 0{ } m } {
4768   \__stex_statements_definiendum_args:n { #1 }
4769   % TODO: root
4770   \stex_get_symbol:n { #2 }
4771   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4772   \str_set:Nx \l_tmpa_str {
4773     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4774   }
4775   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4776   \rustex_if:TF {
4777     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4778       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4779     }
4780   } {
4781     \exp_args:Nnx \defemph@uri {
4782       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4783     } { \l_stex_get_symbol_uri_str }
4784   }
4785 }
4786 \stex_deactivate_macro:Nn \definame {definition~environments}
4787
4788 \NewDocumentCommand \Definame { 0{ } m } {
4789   \__stex_statements_definiendum_args:n { #1 }
4790   \stex_get_symbol:n { #2 }
4791   \str_set:Nx \l_tmpa_str {
4792     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4793   }
4794   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4795   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4796   \rustex_if:TF {

```

```

4797 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4798 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4799 }
4800 } {
4801 \exp_args:Nnx \defemph@uri {
4802 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4803 } { \l_stex_get_symbol_uri_str }
4804 }
4805 }
4806 \stex_deactivate_macro:Nn \Definame {definition~environments}
4807
4808 \NewDocumentCommand \premise { m }{
4809 \stex_annotate:nnn{ premise }{}{ #1 }
4810 }
4811 \NewDocumentCommand \conclusion { m }{
4812 \stex_annotate:nnn{ conclusion }{}{ #1 }
4813 }
4814 \NewDocumentCommand \definiens { O{} m }{
4815 \str_clear:N \l_stex_get_symbol_uri_str
4816 \tl_if_empty:nF {#1} {
4817 \stex_get_symbol:n { #1 }
4818 }
4819 \str_if_empty:NT \l_stex_get_symbol_uri_str {
4820 \int_compare:nNnTF {\clist_count:N \l__stex_statements_sdefinition_for_clist} = 1 {
4821 \str_set:Nx \l_stex_get_symbol_uri_str {\clist_item:Nn \l__stex_statements_sdefinition
4822 }{
4823 % TODO throw error
4824 }
4825 }
4826 \str_if_eq:eeT {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{module}}{
4827 {\l_stex_current_module_str}{
4828 \str_if_eq:eeF {\prop_item:cn {l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defin
4829 {true}}{
4830 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4831 \exp_args:Nx \stex_add_to_current_module:n {
4832 \prop_put:cnn{l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop}{defined}{true}
4833 }
4834 }
4835 }
4836 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4837 }
4838
4839 \stex_deactivate_macro:Nn \premise {definition,~example~or~assertion~environments}
4840 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4841 \stex_deactivate_macro:Nn \definiens {definition~environments}
4842

```

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

sdefinition

```

4843
4844 \keys_define:nn {stex / sdefinition }{
4845 type .str_set_x:N = \sdefinitiontype,
4846 id .str_set_x:N = \sdefinitionid,

```

```

4847 name .str_set_x:N = \sdefinitionname,
4848 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4849 title .tl_set:N = \sdefinitiontitle
4850 }
4851 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4852 \str_clear:N \sdefinitiontype
4853 \str_clear:N \sdefinitionid
4854 \str_clear:N \sdefinitionname
4855 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4856 \tl_clear:N \sdefinitiontitle
4857 \keys_set:nn { stex / sdefinition }{ #1 }
4858 }
4859
4860 \NewDocumentEnvironment{sdefinition}{0{}}{
4861 \__stex_statements_sdefinition_args:n{ #1 }
4862 \stex_reactivate_macro:N \definiendum
4863 \stex_reactivate_macro:N \definame
4864 \stex_reactivate_macro:N \Definame
4865 \stex_reactivate_macro:N \premise
4866 \stex_reactivate_macro:N \definiens
4867 \stex_if_smsmode:F{
4868 \seq_clear:N \l_tmpa_seq
4869 \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4870 \tl_if_empty:nF{ ##1 }{
4871 \stex_get_symbol:n { ##1 }
4872 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4873 \l_stex_get_symbol_uri_str
4874 }
4875 }
4876 }
4877 \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4878 \exp_args:Nnnx
4879 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4880 \str_if_empty:NF \sdefinitiontype {
4881 \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4882 }
4883 \str_if_empty:NF \sdefinitionname {
4884 \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4885 }
4886 \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4887 \tl_clear:N \l_tmpa_tl
4888 \clist_map_inline:Nn \l_tmpa_clist {
4889 \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4890 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4891 }
4892 }
4893 \tl_if_empty:NTF \l_tmpa_tl {
4894 \__stex_statements_sdefinition_start:
4895 }{
4896 \l_tmpa_tl
4897 }
4898 }
4899 \stex_ref_new_doc_target:n \sdefinitionid
4900 \stex_smsmode_do:

```

```

4901 }{
4902   \stex_suppress_html:n {
4903     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4904   }
4905   \stex_if_smsmode:F {
4906     \clist_set:Nn \l_tmpa_clist \sdefinitiontype
4907     \tl_clear:N \l_tmpa_tl
4908     \clist_map_inline:Nn \l_tmpa_clist {
4909       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4910         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4911       }
4912     }
4913     \tl_if_empty:NTF \l_tmpa_tl {
4914       \__stex_statements_sdefinition_end:
4915     }{
4916       \l_tmpa_tl
4917     }
4918     \end{stex_annotate_env}
4919   }
4920 }

```

\stexpatchdefinition

```

4921 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
4922   \par\noindent\titllemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4923     ~(\sdefinitiontitle)
4924   }~}
4925 }
4926 \cs_new_protected:Nn \__stex_statements_sdefinition_end: { \par\medskip}
4927
4928 \newcommand\stexpatchdefinition[3] [] {
4929   \str_set:Nx \l_tmpa_str{ #1 }
4930   \str_if_empty:NTF \l_tmpa_str {
4931     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4932     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4933   }{
4934     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4935     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4936   }
4937 }

```

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

\inlinedef inline:

```

4938 \keys_define:nn {stex / inlinedef }{
4939   type      .str_set_x:N = \sdefinitiontype,
4940   id        .str_set_x:N = \sdefinitionid,
4941   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4942   name      .str_set_x:N = \sdefinitionname
4943 }
4944 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4945   \str_clear:N \sdefinitiontype
4946   \str_clear:N \sdefinitionid
4947   \str_clear:N \sdefinitionname
4948   \clist_clear:N \l__stex_statements_sdefinition_for_clist

```

```

4949 \keys_set:nn { stex / inlinedef }{ #1 }
4950 }
4951 \NewDocumentCommand \inlinedef { 0{} m } {
4952   \beginingroup
4953   \__stex_statements_inlinedef_args:n{ #1 }
4954   \stex_reactivate_macro:N \definiendum
4955   \stex_reactivate_macro:N \definame
4956   \stex_reactivate_macro:N \Definame
4957   \stex_reactivate_macro:N \premise
4958   \stex_reactivate_macro:N \definiens
4959   \stex_ref_new_doc_target:n \sdefinitionid
4960   \stex_if_smsmode:TF{\stex_suppress_html:n {
4961     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4962   }}{
4963     \seq_clear:N \l_tmpa_seq
4964     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4965       \tl_if_empty:NF{ ##1 }{
4966         \stex_get_symbol:n { ##1 }
4967         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4968           \l_stex_get_symbol_uri_str
4969         }
4970       }
4971     }
4972     \clist_set_from_seq:NN \l__stex_statements_sdefinition_for_clist \l_tmpa_seq
4973     \exp_args:Nnx
4974     \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4975       \str_if_empty:NF \sdefinitiontype {
4976         \stex_annotate_invisible:nnn{typestrings}{\sdefinitiontype}{}
4977       }
4978       #2
4979       \str_if_empty:NF \sdefinitionname {
4980         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sdefinitionname}}
4981         \stex_annotate_invisible:nnn{statementname}{\sdefinitionname}{}
4982       }
4983     }
4984   }
4985   \endgroup
4986   \stex_smsmode_do:
4987 }

```

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

32.2 Assertions

sassertion

```

4988 \keys_define:nn {stex / sassertion }{
4989   type      .str_set_x:N = \sassertiontype,
4990   id        .str_set_x:N = \sassertionid,
4991   title     .tl_set:N    = \sassertiontitle ,
4992   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4993   name      .str_set_x:N = \sassertionname
4994 }
4995 }

```

```

4996 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4997   \str_clear:N \sassertiontype
4998   \str_clear:N \sassertionid
4999   \str_clear:N \sassertionname
5000   \clist_clear:N \l__stex_statements_sassertion_for_clist
5001   \tl_clear:N \sassertiontitle
5002   \keys_set:nn { stex / sassertion }{ #1 }
5003 }
5004
5005 %\tl_new:N \g__stex_statements_aftergroup_tl
5006
5007 \NewDocumentEnvironment{sassertion}{0{}}{
5008   \__stex_statements_sassertion_args:n{ #1 }
5009   \stex_reactivate_macro:N \premise
5010   \stex_reactivate_macro:N \conclusion
5011   \stex_if_smsmode:F {
5012     \seq_clear:N \l_tmpa_seq
5013     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5014       \tl_if_empty:nF{ ##1 }{
5015         \stex_get_symbol:n { ##1 }
5016         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5017           \l_stex_get_symbol_uri_str
5018         }
5019       }
5020     }
5021     \exp_args:Nnnx
5022     \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}
5023     \str_if_empty:NF \sassertiontype {
5024       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
5025     }
5026     \str_if_empty:NF \sassertionname {
5027       \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5028     }
5029     \clist_set:Nn \l_tmpa_clist \sassertiontype
5030     \tl_clear:N \l_tmpa_tl
5031     \clist_map_inline:Nn \l_tmpa_clist {
5032       \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
5033         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
5034       }
5035     }
5036     \tl_if_empty:NTF \l_tmpa_tl {
5037       \__stex_statements_sassertion_start:
5038     }{
5039       \l_tmpa_tl
5040     }
5041   }
5042   \str_if_empty:NTF \sassertionid {
5043     \str_if_empty:NF \sassertionname {
5044       \stex_ref_new_doc_target:n {}
5045     }
5046   } {
5047     \stex_ref_new_doc_target:n \sassertionid
5048   }
5049   \stex_smsmode_do:

```

```

5050 }{
5051   \str_if_empty:NF \sassertionname {
5052     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sassertionname}}
5053     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5054   }
5055   \stex_if_smsmode:F {
5056     \clist_set:Nn \l_tmpa_clist \sassertiontype
5057     \tl_clear:N \l_tmpa_tl
5058     \clist_map_inline:Nn \l_tmpa_clist {
5059       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
5060         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
5061       }
5062     }
5063     \tl_if_empty:NTF \l_tmpa_tl {
5064       \__stex_statements_sassertion_end:
5065     }{
5066       \l_tmpa_tl
5067     }
5068     \end{stex_annotate_env}
5069   }
5070 }

```

\stexpatchassertion

```

5071
5072 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
5073   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
5074     (\sassertiontitle)
5075   }~}
5076 }
5077 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
5078
5079 \newcommand\stexpatchassertion[3] [] {
5080   \str_set:Nx \l_tmpa_str{ #1 }
5081   \str_if_empty:NTF \l_tmpa_str {
5082     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
5083     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
5084   }{
5085     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
5086     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
5087   }
5088 }

```

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

\inlineass inline:

```

5089 \keys_define:nn {stex / inlineass }{
5090   type .str_set_x:N = \sassertiontype,
5091   id .str_set_x:N = \sassertionid,
5092   for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
5093   name .str_set_x:N = \sassertionname
5094 }
5095 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
5096   \str_clear:N \sassertiontype
5097   \str_clear:N \sassertionid

```

```

5098 \str_clear:N \sassertionname
5099 \clist_clear:N \l__stex_statements_sassertion_for_clist
5100 \keys_set:nn { stex / inlineass }{ #1 }
5101 }
5102 \NewDocumentCommand \inlineass { 0{} m } {
5103 \beginngroup
5104 \stex_reactivate_macro:N \premise
5105 \stex_reactivate_macro:N \conclusion
5106 \__stex_statements_inlineass_args:n{ #1 }
5107 \str_if_empty:NTF \sassertionid {
5108 \str_if_empty:NF \sassertionname {
5109 \stex_ref_new_doc_target:n {}
5110 }
5111 } {
5112 \stex_ref_new_doc_target:n \sassertionid
5113 }
5114
5115 \stex_if_smsmode:TF{
5116 \str_if_empty:NF \sassertionname {
5117 \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5118 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5119 }
5120 }{
5121 \seq_clear:N \l_tmpa_seq
5122 \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
5123 \tl_if_empty:NF{ ##1 }{
5124 \stex_get_symbol:n { ##1 }
5125 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5126 \l_stex_get_symbol_uri_str
5127 }
5128 }
5129 }
5130 \exp_args:Nnx
5131 \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
5132 \str_if_empty:NF \sassertiontype {
5133 \stex_annotate_invisible:nnn{typestrings}{\sassertiontype}{}
5134 }
5135 #2
5136 \str_if_empty:NF \sassertionname {
5137 \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sassertionname}}
5138 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
5139 \stex_annotate_invisible:nnn{statementname}{\sassertionname}{}
5140 }
5141 }
5142 }
5143 \endgroup
5144 \stex_smsmode_do:
5145 }

```

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

32.3 Examples

sexample

```

5146 \keys_define:nn {stex / sexample }{
5147   type      .str_set_x:N = \exampletype,
5148   id        .str_set_x:N = \sexampleid,
5149   title     .tl_set:N     = \sexamplename,
5150   name      .str_set_x:N = \sexamplename ,
5151   for       .clist_set:N = \l__stex_statements_sexample_for_clist,
5152 }
5153 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
5154   \str_clear:N \sexampletype
5155   \str_clear:N \sexampleid
5156   \str_clear:N \sexamplename
5157   \tl_clear:N \sexamplename
5158   \clist_clear:N \l__stex_statements_sexample_for_clist
5159   \keys_set:nn { stex / sexample }{ #1 }
5160 }
5161
5162 \NewDocumentEnvironment{sexample}{0{}}{
5163   \__stex_statements_sexample_args:n{ #1 }
5164   \stex_reactivate_macro:N \premise
5165   \stex_reactivate_macro:N \conclusion
5166   \stex_if_smsmode:F {
5167     \seq_clear:N \l_tmpa_seq
5168     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5169       \tl_if_empty:NF{ ##1 }{
5170         \stex_get_symbol:n { ##1 }
5171         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5172           \l_stex_get_symbol_uri_str
5173         }
5174       }
5175     }
5176   }
5177   \exp_args:Nnnx
5178   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5179   \str_if_empty:NF \sexampletype {
5180     \stex_annotate_invisible:nnn{typestrings}{\sexampletype}{}
5181   }
5182   \str_if_empty:NF \sexamplename {
5183     \stex_annotate_invisible:nnn{statementname}{\sexamplename}{}
5184   }
5185   \clist_set:N \l_tmpa_clist \sexampletype
5186   \tl_clear:N \l_tmpa_tl
5187   \clist_map_inline:Nn \l_tmpa_clist {
5188     \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5189       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5190     }
5191   }
5192   \tl_if_empty:NTF \l_tmpa_tl {
5193     \__stex_statements_sexample_start:
5194   }{
5195     \l_tmpa_tl
5196   }

```

```

5197 }
5198 \str_if_empty:NF \sexampleid {
5199   \stex_ref_new_doc_target:n \sexampleid
5200 }
5201 \stex_smsmode_do:
5202 ){
5203   \str_if_empty:NF \sexamplename {
5204     \stex_suppress_html:n{\stex_symdecl_do:nn}{\sexamplename}}
5205   }
5206   \stex_if_smsmode:F {
5207     \clist_set:Nn \l_tmpa_clist \sexamplotype
5208     \tl_clear:N \l_tmpa_tl
5209     \clist_map_inline:Nn \l_tmpa_clist {
5210       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5211         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5212       }
5213     }
5214     \tl_if_empty:NTF \l_tmpa_tl {
5215       \__stex_statements_sexample_end:
5216     }{
5217       \l_tmpa_tl
5218     }
5219     \end{stex_annotate_env}
5220   }
5221 }

```

\stexpatchexample

```

5222
5223 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5224   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
5225     (\sexamplotype)
5226   }~}
5227 }
5228 \cs_new_protected:Nn \__stex_statements_sexample_end: { \par\medskip}
5229
5230 \newcommand\stexpatchexample[3]{} {
5231   \str_set:Nx \l_tmpa_str{ #1 }
5232   \str_if_empty:NTF \l_tmpa_str {
5233     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5234     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5235   }{
5236     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5237     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5238   }
5239 }

```

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

\inlineex inline:

```

5240 \keys_define:nn {stex / inlineex }{
5241   type      .str_set_x:N = \sexamplotype,
5242   id        .str_set_x:N = \sexampleid,
5243   for       .clist_set:N = \l__stex_statements_sexample_for_clist ,
5244   name      .str_set_x:N = \sexamplename

```

```

5245 }
5246 \cs_new_protected:Nn \__stex_statements_inlineex_args:n {
5247   \str_clear:N \sexamplotype
5248   \str_clear:N \sexampleid
5249   \str_clear:N \sexamplename
5250   \clist_clear:N \l__stex_statements_sexample_for_clist
5251   \keys_set:nn { stex / inlineex }{ #1 }
5252 }
5253 \NewDocumentCommand \inlineex { 0{ } m } {
5254   \beginngroup
5255   \stex_reactivate_macro:N \premise
5256   \stex_reactivate_macro:N \conclusion
5257   \__stex_statements_inlineex_args:n{ #1 }
5258   \str_if_empty:NF \sexampleid {
5259     \stex_ref_new_doc_target:n \sexampleid
5260   }
5261   \stex_if_smsmode:TF{
5262     \str_if_empty:NF \sexamplename {
5263       \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5264     }
5265   }{
5266     \seq_clear:N \l_tmpa_seq
5267     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5268       \tl_if_empty:NF{ ##1 }{
5269         \stex_get_symbol:n { ##1 }
5270         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5271           \l_stex_get_symbol_uri_str
5272         }
5273       }
5274     }
5275     \exp_args:Nnx
5276     \stex_annotate:nnn{example}{\seq_use:Nn \l_tmpa_seq {,}}{
5277       \str_if_empty:NF \sexamplotype {
5278         \stex_annotate_invisible:nnn{typestrings}{\sexamplotype}{ }
5279       }
5280       #2
5281       \str_if_empty:NF \sexamplename {
5282         \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sexamplename}}
5283         \stex_annotate_invisible:nnn{statementname}{\sexamplename}{ }
5284       }
5285     }
5286   }
5287   \endgroup
5288   \stex_smsmode_do:
5289 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5290 \keys_define:nn { stex / sparagraph } {
5291   id          .str_set_x:N    = \sparagraphid ,

```

```

5292 title .tl_set:N = \l_stex_sparagraph_title_tl ,
5293 type .str_set_x:N = \sparagraphtype ,
5294 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5295 from .tl_set:N = \sparagraphfrom ,
5296 to .tl_set:N = \sparagraphto ,
5297 start .tl_set:N = \l_stex_sparagraph_start_tl ,
5298 name .str_set:N = \sparagraphname
5299 }
5300
5301 \cs_new_protected:Nn \stex_sparagraph_args:n {
5302 \tl_clear:N \l_stex_sparagraph_title_tl
5303 \tl_clear:N \sparagraphfrom
5304 \tl_clear:N \sparagraphto
5305 \tl_clear:N \l_stex_sparagraph_start_tl
5306 \str_clear:N \sparagraphid
5307 \str_clear:N \sparagraphtype
5308 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5309 \str_clear:N \sparagraphname
5310 \keys_set:nn { stex / sparagraph }{ #1 }
5311 }
5312 \newif\if@in@omtext\@in@omtextfalse
5313
5314 \NewDocumentEnvironment {sparagraph} { 0{} } {
5315 \stex_sparagraph_args:n { #1 }
5316 \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5317 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5318 }{
5319 \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5320 }
5321 \@in@omtexttrue
5322 \stex_if_smsmode:F {
5323 \seq_clear:N \l_tmpa_seq
5324 \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5325 \tl_if_empty:NF{ ##1 }{
5326 \stex_get_symbol:n { ##1 }
5327 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5328 \l_stex_get_symbol_uri_str
5329 }
5330 }
5331 }
5332 \exp_args:Nnnx
5333 \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5334 \str_if_empty:NF \sparagraphtype {
5335 \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{ }
5336 }
5337 \str_if_empty:NF \sparagraphfrom {
5338 \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{ }
5339 }
5340 \str_if_empty:NF \sparagraphto {
5341 \stex_annotate_invisible:nnn{to}{\sparagraphto}{ }
5342 }
5343 \str_if_empty:NF \sparagraphname {
5344 \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{ }
5345 }

```

```

5346 \clist_set:No \l_tmpa_clist \sparagraphtype
5347 \tl_clear:N \l_tmpa_tl
5348 \clist_map_inline:Nn \sparagraphtype {
5349   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5350     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5351   }
5352 }
5353 \tl_if_empty:NTF \l_tmpa_tl {
5354   \__stex_statements_sparagraph_start:
5355 }{
5356   \l_tmpa_tl
5357 }
5358 }
5359 \clist_set:No \l_tmpa_clist \sparagraphtype
5360 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5361 {
5362   \stex_reactivate_macro:N \definiendum
5363   \stex_reactivate_macro:N \definame
5364   \stex_reactivate_macro:N \Definame
5365   \stex_reactivate_macro:N \premise
5366   \stex_reactivate_macro:N \definens
5367 }
5368 \str_if_empty:NTF \sparagraphid {
5369   \str_if_empty:NTF \sparagraphname {
5370     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5371       \stex_ref_new_doc_target:n {}
5372     }
5373   } {
5374     \stex_ref_new_doc_target:n {}
5375   }
5376 } {
5377   \stex_ref_new_doc_target:n \sparagraphid
5378 }
5379 \exp_args:NNx
5380 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5381   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5382     \tl_if_empty:nF{ ##1 }{
5383       \stex_get_symbol:n { ##1 }
5384       \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5385     }
5386   }
5387 }
5388 \stex_smsmode_do:
5389 \ignorespacesandpars
5390 }{
5391   \str_if_empty:NF \sparagraphname {
5392     \stex_suppress_html:n{\stex_symdecl_do:nn{}{\sparagraphname}}
5393     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5394   }
5395   \stex_if_smsmode:F {
5396     \clist_set:No \l_tmpa_clist \sparagraphtype
5397     \tl_clear:N \l_tmpa_tl
5398     \clist_map_inline:Nn \l_tmpa_clist {
5399       \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{

```

```

5400     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5401   }
5402 }
5403 \tl_if_empty:NTF \l_tmpa_tl {
5404   \__stex_statements_sparagraph_end:
5405 }{
5406   \l_tmpa_tl
5407 }
5408 \end{stex_annotate_env}
5409 }
5410 }

```

\stexpatchparagraph

```

5411
5412 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5413   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5414     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5415       \titleemph{\l_stex_sparagraph_title_tl}:~
5416     }
5417   }{
5418     \titleemph{\l_stex_sparagraph_start_tl}~
5419   }
5420 }
5421 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5422
5423 \newcommand\stexpatchparagraph[3] [] {
5424   \str_set:Nx \l_tmpa_str{ #1 }
5425   \str_if_empty:NTF \l_tmpa_str {
5426     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5427     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5428   }{
5429     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
5430     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5431   }
5432 }
5433
5434 \keys_define:nn { stex / inlinepara } {
5435   id      .str_set_x:N = \sparagraphid ,
5436   type    .str_set_x:N = \sparagraphtype ,
5437   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5438   from    .tl_set:N    = \sparagraphfrom ,
5439   to      .tl_set:N    = \sparagraphto ,
5440   name    .str_set:N   = \sparagraphname
5441 }
5442 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5443   \tl_clear:N \sparagraphfrom
5444   \tl_clear:N \sparagraphto
5445   \str_clear:N \sparagraphid
5446   \str_clear:N \sparagraphtype
5447   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5448   \str_clear:N \sparagraphname
5449   \keys_set:nn { stex / inlinepara }{ #1 }
5450 }
5451 \NewDocumentCommand \inlinepara { 0{} m } {

```

```

5452 \begingroup
5453 \__stex_statements_inlinepara_args:n{ #1 }
5454 \clist_set:No \l_tmpa_clist \sparagraphtype
5455 \str_if_empty:NTF \sparagraphid {
5456   \str_if_empty:NTF \sparagraphname {
5457     \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5458       \stex_ref_new_doc_target:n {}
5459     }
5460   } {
5461     \stex_ref_new_doc_target:n {}
5462   }
5463 } {
5464   \stex_ref_new_doc_target:n \sparagraphid
5465 }
5466 \stex_if_smsmode:TF{
5467   \str_if_empty:NF \sparagraphname {
5468     \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5469   \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5470 }
5471 }{
5472   \seq_clear:N \l_tmpa_seq
5473   \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5474     \tl_if_empty:nF{ ##1 }{
5475       \stex_get_symbol:n { ##1 }
5476       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5477         \l_stex_get_symbol_uri_str
5478       }
5479     }
5480   }
5481   \exp_args:Nnx
5482   \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5483     \str_if_empty:NF \sparagraphtype {
5484       \stex_annotate_invisible:nnn{typestrings}{\sparagraphtype}{}
5485     }
5486     \str_if_empty:NF \sparagraphfrom {
5487       \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5488     }
5489     \str_if_empty:NF \sparagraphto {
5490       \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5491     }
5492     \str_if_empty:NF \sparagraphname {
5493       \stex_suppress_html:n{\stex_symdecl_do:nn{}}{\sparagraphname}}
5494     \stex_annotate_invisible:nnn{statementname}{\sparagraphname}{}
5495     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5496   }
5497   \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5498     \clist_map_inline:Nn \l_tmpa_seq {
5499       \stex_ref_new_sym_target:n {##1}
5500     }
5501   }
5502   #2
5503 }
5504 }
5505 \endgroup

```

```

5506 \stex_smsmode_do:
5507 }
5508
(End definition for \stexpatchparagraph. This function is documented on page 42.)
5509 </package>

```


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

```
5510 <*package>
5511 <@@=stex_sproof>
5512
5513 %%%%%%%%%% sproof.dtx %%%%%%%%%%
5514
```

33.2 Proofs

We first define some keys for the proof environment.

```
5515 \keys_define:nn { stex / spf } {
5516   id          .str_set_x:N = \spfid,
5517   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5518   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5519   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5520   type        .str_set_x:N = \spftype,
5521   title       .tl_set:N    = \spftitle,
5522   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5523   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5524   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5525 }
5526 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5527   \str_clear:N \spfid
5528   \tl_clear:N \l__stex_sproof_spf_for_tl
5529   \tl_clear:N \l__stex_sproof_spf_from_tl
5530   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5531   \str_clear:N \spftype
5532   \tl_clear:N \spftitle
5533   \tl_clear:N \l__stex_sproof_spf_continues_tl
5534   \tl_clear:N \l__stex_sproof_spf_functions_tl
```

⁸EdNOTE: need an implementation for L^AT_EX_ML

```

5535 \tl_clear:N \l__stex_sproof_spf_method_tl
5536 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5537 \keys_set:nn { stex / spf }{ #1 }
5538 }

```

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

```

5539 \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.

```

5540 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5541 \cs_new_protected:Npn \sproofnumber {
5542   \int_set:Nn \l_tmpa_int {1}
5543   \bool_while_do:nn {
5544     \int_compare_p:nNn {
5545       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5546     } > 0
5547   }{
5548     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5549     \int_incr:N \l_tmpa_int
5550   }
5551 }
5552 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5553   \int_set:Nn \l_tmpa_int {1}
5554   \bool_while_do:nn {
5555     \int_compare_p:nNn {
5556       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5557     } > 0
5558   }{
5559     \int_incr:N \l_tmpa_int
5560   }
5561   \int_compare:nNnF \l_tmpa_int = 1 {
5562     \int_decr:N \l_tmpa_int
5563   }
5564   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5565     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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

```

5566 }
5567 }
5568
5569 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5570   \int_set:Nn \l_tmpa_int {1}
5571   \bool_while_do:nn {
5572     \int_compare_p:nNn {
5573       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5574     } > 0
5575   }{
5576     \int_incr:N \l_tmpa_int
5577   }
5578   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5579 }
5580
5581 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5582   \int_set:Nn \l_tmpa_int {1}
5583   \bool_while_do:nn {
5584     \int_compare_p:nNn {
5585       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5586     } > 0
5587   }{
5588     \int_incr:N \l_tmpa_int
5589   }
5590   \int_decr:N \l_tmpa_int
5591   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5592 }

```

\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

```

5593 \def\sproof@box{
5594   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5595 }
5596 \def\sproofend{
5597   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5598     \hfil\hfill\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5599   }
5600 }

```

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

spf@*@kw

```

5601 \def\spf@proofsketch@kw{Proof-Sketch}
5602 \def\spf@proof@kw{Proof}
5603 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5604 \AddToHook{begindocument}{
5605   \ltx@ifpackageloaded{babel}{
5606     \makeatletter
5607     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5608     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5609       \input{sproof-ngerman.ldf}

```

```

5610 }
5611 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5612   \input{sproof-finnish.ldf}
5613 }
5614 \clist_if_in:NnT \l_tmpa_clist {french}{
5615   \input{sproof-french.ldf}
5616 }
5617 \clist_if_in:NnT \l_tmpa_clist {russian}{
5618   \input{sproof-russian.ldf}
5619 }
5620 \makeatother
5621 }{}
5622 }

```

spfsketch

```

5623 \newcommand\spfsketch[2] [] {
5624   \beginingroup
5625   \let \premise \stex_proof_premise:
5626   \__stex_sproof_spf_args:n{#1}
5627   \stex_if_smsmode:TF {
5628     \str_if_empty:NF \spfid {
5629       \stex_ref_new_doc_target:n \spfid
5630     }
5631   }{
5632     \seq_clear:N \l_tmpa_seq
5633     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5634       \tl_if_empty:nF{ ##1 }{
5635         \stex_get_symbol:n { ##1 }
5636         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5637           \l_stex_get_symbol_uri_str
5638         }
5639       }
5640     }
5641     \exp_args:Nnx
5642     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5643       \str_if_empty:NF \spftype {
5644         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5645       }
5646       \clist_set:No \l_tmpa_clist \spftype
5647       \tl_set:Nn \l_tmpa_tl {
5648         \titleemph{
5649           \tl_if_empty:NTF \spftitle {
5650             \spf@proofsketch@kw
5651           }{
5652             \spftitle
5653           }
5654         }::~
5655       }
5656       \clist_map_inline:Nn \l_tmpa_clist {
5657         \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5658           \tl_clear:N \l_tmpa_tl
5659         }
5660       }
5661       \str_if_empty:NF \spfid {

```

```

5662         \stex_ref_new_doc_target:n \spfid
5663     }
5664     \l_tmpa_tl #2 \sproofend
5665 }
5666 }
5667 \endgroup
5668 \stex_smsmode_do:
5669 }
5670

```

(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

```

5671 \newenvironment{spfeq}[2][]{
5672   \__stex_sproof_spf_args:n{#1}
5673   \let \premise \stex_proof_premise:
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:Nnnx
5689     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5690     \str_if_empty:NF \spftype {
5691       \stex_annotate_invisible:nnn{type}{\spftype}{}
5692     }
5693
5694     \clist_set:No \l_tmpa_clist \spftype
5695     \tl_clear:N \l_tmpa_tl
5696     \clist_map_inline:Nn \l_tmpa_clist {
5697       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5698         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5699       }
5700       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5701         \tl_set:Nn \l_tmpa_tl {\use:n{}}
5702       }
5703     }
5704     \tl_if_empty:NTF \l_tmpa_tl {
5705       \__stex_sproof_spfeq_start:
5706     }{
5707       \l_tmpa_tl
5708     }{-#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

```

5709 \str_if_empty:NF \spfid {
5710 \stex_ref_new_doc_target:n \spfid
5711 }
5712 \begin{displaymath}\begin{array}{rcll}
5713 }
5714 \stex_smsmode_do:
5715 }{
5716 \stex_if_smsmode:F {
5717 \end{array}\end{displaymath}
5718 \clist_set:No \l_tmpa_clist \spftype
5719 \tl_clear:N \l_tmpa_tl
5720 \clist_map_inline:Nn \l_tmpa_clist {
5721 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5722 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5723 }
5724 }
5725 \tl_if_empty:NTF \l_tmpa_tl {
5726 \__stex_sproof_spfeq_end:
5727 }{
5728 \l_tmpa_tl
5729 }
5730 \end{stex_annotate_env}
5731 }
5732 }
5733
5734 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5735 \titleemph{
5736 \tl_if_empty:NTF \spftitle {
5737 \spf@proof@kw
5738 }{
5739 \spftitle
5740 }
5741 }:
5742 }
5743 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5744
5745 \newcommand\stexpatchspfeq[3] [] {
5746 \str_set:Nx \l_tmpa_str{ #1 }
5747 \str_if_empty:NTF \l_tmpa_str {
5748 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5749 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5750 }{
5751 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5752 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5753 }
5754 }
5755

```

(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.

```

5756 \newenvironment{sproof}[2] []{

```

```

5757 \let \premise \stex_proof_premise:
5758 \intarray_gzero:N \l__stex_sproof_counter_intarray
5759 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5760 \__stex_sproof_spf_args:n{#1}
5761 \stex_if_smsmode:TF {
5762   \str_if_empty:NF \spfid {
5763     \stex_ref_new_doc_target:n \spfid
5764   }
5765 }{
5766   \seq_clear:N \l_tmpa_seq
5767   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5768     \tl_if_empty:NF{ ##1 }{
5769       \stex_get_symbol:n { ##1 }
5770       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5771         \l_stex_get_symbol_uri_str
5772       }
5773     }
5774   }
5775   \exp_args:Nnnx
5776   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5777   \str_if_empty:NF \spftype {
5778     \stex_annotate_invisible:nnn{type}{\spftype}{}
5779   }
5780
5781   \clist_set:No \l_tmpa_clist \spftype
5782   \tl_clear:N \l_tmpa_tl
5783   \clist_map_inline:Nn \l_tmpa_clist {
5784     \tl_if_exist:cT {\__stex_sproof_sproof_##1_start:}{
5785       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_start:}}
5786     }
5787     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5788       \tl_set:Nn \l_tmpa_tl {\use:n{}}
5789     }
5790   }
5791   \tl_if_empty:NTF \l_tmpa_tl {
5792     \__stex_sproof_sproof_start:
5793   }{
5794     \l_tmpa_tl
5795   }{~#2}
5796   \str_if_empty:NF \spfid {
5797     \stex_ref_new_doc_target:n \spfid
5798   }
5799   \begin{description}
5800 }
5801 \stex_smsmode_do:
5802 }{
5803   \stex_if_smsmode:F{
5804     \end{description}
5805     \clist_set:No \l_tmpa_clist \spftype
5806     \tl_clear:N \l_tmpa_tl
5807     \clist_map_inline:Nn \l_tmpa_clist {
5808       \tl_if_exist:cT {\__stex_sproof_sproof_##1_end:}{
5809         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_sproof_sproof_##1_end:}}
5810       }

```

```

5811     }
5812     \tl_if_empty:NTF \l_tmpa_tl {
5813       \__stex_sproof_sproof_end:
5814     }{
5815       \l_tmpa_tl
5816     }
5817     \end{stex_annotate_env}
5818   }
5819 }
5820
5821 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5822   \par\noindent\titleemph{
5823     \tl_if_empty:NTF \spftype {
5824       \spf@proof@kw
5825     }{
5826       \spftype
5827     }
5828   }:
5829 }
5830 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5831
5832 \newcommand\stexpatchproof[3] [] {
5833   \str_set:Nx \l_tmpa_str{ #1 }
5834   \str_if_empty:NTF \l_tmpa_str {
5835     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5836     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5837   }{
5838     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5839     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5840   }
5841 }

```

\spfidea

```

5842 \newcommand\spfidea[2] []{
5843   \__stex_sproof_spf_args:n{#1}
5844   \titleemph{
5845     \tl_if_empty:NTF \spftype {Proof~Idea}{
5846       \spftype
5847     }:
5848   }~#2
5849   \sproofend
5850 }

```

(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

```

5851 \newenvironment{spfstep}[1] []{
5852   \__stex_sproof_spf_args:n{#1}
5853   \stex_if_smsmode:TF {

```



```

5854 \str_if_empty:NF \spfid {
5855 \stex_ref_new_doc_target:n \spfid
5856 }
5857 }{
5858 \@in@omtexttrue
5859 \seq_clear:N \l_tmpa_seq
5860 \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5861 \tl_if_empty:nF{ ##1 }{
5862 \stex_get_symbol:n { ##1 }
5863 \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5864 \l_stex_get_symbol_uri_str
5865 }
5866 }
5867 }
5868 \exp_args:Nnnx
5869 \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5870 \str_if_empty:NF \spftype {
5871 \stex_annotate_invisible:nnn{type}{\spftype}{}
5872 }
5873 \clist_set:No \l_tmpa_clist \spftype
5874 \tl_set:Nn \l_tmpa_tl {
5875 \item[\sproofnumber]
5876 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5877 }
5878 \clist_map_inline:Nn \l_tmpa_clist {
5879 \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5880 \tl_clear:N \l_tmpa_tl
5881 }
5882 }
5883 \l_tmpa_tl
5884 \tl_if_empty:NF \spftitle {
5885 {(\titleemph{\spftitle})\enspace}
5886 }
5887 \str_if_empty:NF \spfid {
5888 \stex_ref_new_doc_target:n \spfid
5889 }
5890 }
5891 \stex_smsmode_do:
5892 \ignorespacesandpars
5893 }{
5894 \bool_if:NT \l__stex_sproof_inc_counter_bool {
5895 \__stex_sproof_inc_counter:
5896 }
5897 \stex_if_smsmode:F {
5898 \end{stex_annotate_env}
5899 }
5900 }

```

sproofcomment

```

5901 \newenvironment{sproofcomment}[1][]{
5902 \__stex_sproof_spf_args:n{#1}
5903 \clist_set:No \l_tmpa_clist \spftype
5904 \tl_set:Nn \l_tmpa_tl {
5905 \item[\sproofnumber]

```

```

5906 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5907 }
5908 \clist_map_inline:Nn \l_tmpa_clist {
5909   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5910     \tl_clear:N \l_tmpa_tl
5911   }
5912 }
5913 \l_tmpa_tl
5914 }{
5915   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5916     \__stex_sproof_inc_counter:
5917   }
5918 }

```

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.

```

5919 \newenvironment{subproof}[2][]{
5920   \__stex_sproof_spf_args:n{#1}
5921   \stex_if_smsmode:TF{
5922     \str_if_empty:NF \spfid {
5923       \stex_ref_new_doc_target:n \spfid
5924     }
5925   }{
5926     \seq_clear:N \l_tmpa_seq
5927     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5928       \tl_if_empty:nF{ ##1 }{
5929         \stex_get_symbol:n { ##1 }
5930         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5931           \l_stex_get_symbol_uri_str
5932         }
5933       }
5934     }
5935     \exp_args:Nnnx
5936     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5937     \str_if_empty:NF \spftype {
5938       \stex_annotate_invisible:nnn{type}{\spftype}{\}
5939     }
5940
5941     \clist_set:No \l_tmpa_clist \spftype
5942     \tl_set:Nn \l_tmpa_tl {
5943       \item[\sproofnumber]
5944       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5945     }
5946     \clist_map_inline:Nn \l_tmpa_clist {
5947       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5948         \tl_clear:N \l_tmpa_tl
5949       }
5950     }
5951     \l_tmpa_tl
5952     \tl_if_empty:NF \spftitle {
5953       {(\titleemph{\spftitle})\enspace}
5954     }

```

```

5955     {~#2}
5956     \str_if_empty:NF \spfid {
5957       \stex_ref_new_doc_target:n \spfid
5958     }
5959   }
5960   \__stex_sproof_add_counter:
5961   \stex_smsmode_do:
5962 }{
5963   \__stex_sproof_remove_counter:
5964   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5965     \__stex_sproof_inc_counter:
5966   }
5967   \stex_if_smsmode:F{
5968     \end{stex_annotate_env}
5969   }
5970 }

```

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

```

5971 \newenvironment{spfcases}[2][]{
5972   \tl_if_empty:nTF{#1}{
5973     \begin{subproof}[method=by-cases]{#2}
5974   }{
5975     \begin{subproof}[#1,method=by-cases]{#2}
5976   }
5977 }{
5978   \end{subproof}
5979 }

```

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

```

5980 \newenvironment{spfcase}[2][]{
5981   \__stex_sproof_spf_args:n{#1}
5982   \stex_if_smsmode:TF {
5983     \str_if_empty:NF \spfid {
5984       \stex_ref_new_doc_target:n \spfid
5985     }
5986   }{
5987     \seq_clear:N \l_tmpa_seq
5988     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5989       \tl_if_empty:nF{ ##1 }{
5990         \stex_get_symbol:n { ##1 }
5991         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5992           \l_stex_get_symbol_uri_str
5993         }
5994       }
5995     }
5996     \exp_args:Nnnx
5997     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5998     \str_if_empty:NF \spftype {
5999       \stex_annotate_invisible:nnn{type}{\spftype}{}}
6000   }
6001   \clist_set:Nn \l_tmpa_clist \spftype
6002   \tl_set:Nn \l_tmpa_tl {
6003     \item[\sproofnumber]

```

```

6004     \bool_set_true:N \l__stex_sproof_inc_counter_bool
6005   }
6006   \clist_map_inline:Nn \l_tmpa_clist {
6007     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6008       \tl_clear:N \l_tmpa_tl
6009     }
6010   }
6011   \l_tmpa_tl
6012   \tl_if_empty:nF{#2}{
6013     \titleemph{#2}:~
6014   }
6015 }
6016 \__stex_sproof_add_counter:
6017 \stex_smsmode_do:
6018 ){
6019   \__stex_sproof_remove_counter:
6020   \bool_if:NT \l__stex_sproof_inc_counter_bool {
6021     \__stex_sproof_inc_counter:
6022   }
6023   \stex_if_smsmode:F{
6024     \clist_set:No \l_tmpa_clist \spftype
6025     \tl_set:Nn \l_tmpa_tl{\sproofend}
6026     \clist_map_inline:Nn \l_tmpa_clist {
6027       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
6028         \tl_clear:N \l_tmpa_tl
6029       }
6030     }
6031     \l_tmpa_tl
6032     \end{stex_annotate_env}
6033   }
6034 }

```

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

```

6035 \newcommand\spfcasesketch[3][]{
6036   \begin{spfcase}[#1]{#2}#3\end{spfcase}
6037 }

```

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.

```

6038 \keys_define:nn { stex / just }{
6039   id      .str_set:x:N = \l__stex_sproof_just_id_str,
6040   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
6041   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
6042   args     .tl_set:N    = \l__stex_sproof_just_args_tl
6043 }

```

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

```
6044 \newenvironment{justification}[1] [] {}{}
```

\premise

```
6045 \newcommand\stex_proof_promise:[2] [] {#2}
```

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

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

```
6046 \newcommand\justarg[2] [] {#2}
```

```
6047 \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

```
6048 <*package>
6049
6050 %%%%%%%%%% others.dtx %%%%%%%%%%
6051
6052 <@@=stex_others>
    Warnings and error messages
6053 % None

\MSC Math subject classifier

6054 \NewDocumentCommand \MSC {m} {
6055 % TODO
6056 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
6057 \@ifpackageloaded{tikzinput}{
6058 \RequirePackage{stex-tikzinput}
6059 }{}
6060 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
6061 <*package>
6062 <@@=stex_modules>
6063
6064 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
6065
6066 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
6067 \begingroup
6068 \stex_module_setup:nn{
6069   ns=\c_stex_metatheory_ns_str,
6070   meta=NONE
6071 }{Metatheory}
6072 \stex_reactivate_macro:N \symdecl
6073 \stex_reactivate_macro:N \notation
6074 \stex_reactivate_macro:N \symdef
6075 \ExplSyntaxOff
6076 \csname stex_suppress_html:n\endcsname{
6077   % is-a (a:A, a \in A, a is an A, etc.)
6078   \symdecl{isa}[args=ai]
6079   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
6080   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
6081   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
6082
6083   % bind (\forall, \Pi, \lambda etc.)
6084   \symdecl{bind}[args=Bi]
6085   \notation{bind}[forall]{\comp\forall #1.\;#2}{##1 \comp, ##2}
6086   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
6087   \notation{bind}[depfun]{\comp( #1 \comp{}\;\to\;} #2}{##1 \comp, ##2}
6088
6089   % implicit bind
6090   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1\comp,##2}
6091
6092   % dummy variable
6093   \symdecl{dummyvar}
6094   \notation{dummyvar}[underscore]{\comp\_}
6095   \notation{dummyvar}[dot]{\comp\cdot}
```

```

6096 \notation{dummyvar}[dash]{\comp{\rm --}}
6097
6098 %fromto (function space, Hom-set, implication etc.)
6099 \symdecl{fromto}[args=ai]
6100 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
6101 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
6102
6103 % mapto (lambda etc.)
6104 \symdecl{mapto}[args=Bi]
6105 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
6106 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
6107 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
6108
6109 % function/operator application
6110 \symdecl{apply}[args=ia]
6111 \notation{apply}[prec=0;0x\infprec,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
6112 \notation{apply}[prec=0;0x\infprec,lambda]{#1 \; #2 }{##1 \; ; ##2}
6113
6114 % collection of propositions/booleans/truth values
6115 \symdecl{prop}[name=proposition]
6116 \notation{prop}[prop]{\comp{\rm prop}}
6117 \notation{prop}[BOOL]{\comp{\rm BOOL}}
6118
6119 \symdecl{judgmentholds}[args=1]
6120 \notation{judgmentholds}[vdash,op=\vdash]{\comp\vdash\; ; #1}
6121
6122 % sequences
6123 \symdecl{seqtype}[args=1]
6124 \notation{seqtype}[kleene]{#1^{\comp\ast}}
6125
6126 \symdecl{seqexpr}[args=a]
6127 \notation{seqexpr}[angle,prec=nobrackets]{\comp\angle #1\comp\rangle}{##1\comp,##2}
6128
6129 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
6130 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{#2}}
6131
6132 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
6133 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
6134 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}}{##1\comp,##2}
6135
6136 % letin (''let'', local definitions, variable substitution)
6137 \symdecl{letin}[args=bii]
6138 \notation{letin}[let]{\comp{\rm let}}\;#1\comp{=}\#2\; \comp{\rm in}}\;#3}
6139 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
6140 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
6141
6142 % structures
6143 \symdecl*{module-type}[args=1]
6144 \notation{module-type}{\comp{\mathtt{MOD}}} #1}
6145 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
6146 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
6147
6148 % objects
6149 \symdecl{object}

```



```

6150 \notation{object}{\comp{\mathtt{OBJECT}}}
6151
6152 }
6153 \ExplSyntaxOn
6154 \stex_add_to_current_module:n{
6155   \let\nappa\apply
6156   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
6157   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
6158   \def\livar{\csname sequence-index\endcsname[li]}
6159   \def\uivar{\csname sequence-index\endcsname[ui]}
6160   \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
6161   \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
6162   \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
6163 }
6164 \__stex_modules_end_module:
6165 \endgroup
6166 \</package>

```

Chapter 36

Tikzinput Implementation

```
6167 <*package>
6168
6169 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
6170
6171 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
6172 \RequirePackage{l3keys2e}
6173
6174 \keys_define:nn { tikzinput } {
6175   image .bool_set:N = \c_tikzinput_image_bool,
6176   image .default:n = false ,
6177   unknown .code:n = {}
6178 }
6179
6180 \ProcessKeysOptions { tikzinput }
6181
6182 \bool_if:NTF \c_tikzinput_image_bool {
6183   \RequirePackage{graphicx}
6184
6185   \providecommand\usetikzlibrary[]{}
6186   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
6187 }{
6188   \RequirePackage{tikz}
6189   \RequirePackage{standalone}
6190
6191   \newcommand \tikzinput [2] [] {
6192     \setkeys{Gin}{#1}
6193     \ifx \Gin@ewidth \Gin@exclamation
6194       \ifx \Gin@eheight \Gin@exclamation
6195         \input { #2 }
6196       \else
6197         \resizebox{!}{ \Gin@eheight }{
6198           \input { #2 }
6199         }
6200       \fi
6201     \else
6202       \ifx \Gin@eheight \Gin@exclamation
6203         \resizebox{ \Gin@ewidth }{!}{
6204           \input { #2 }
```

```

6205     }
6206     \else
6207         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6208             \input { #2 }
6209         }
6210     \fi
6211 \fi
6212 }
6213 }
6214
6215 \newcommand \ctikzinput [2] [] {
6216     \begin{center}
6217         \tikzinput [ #1 ] { #2 }
6218     \end{center}
6219 }
6220
6221 \@ifpackageloaded{stex}{
6222     \RequirePackage{stex-tikzinput}
6223 }{}
6224
6225 </package>
6226 <*stex>
6227 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6228 \RequirePackage{stex}
6229 \RequirePackage{tikzinput}
6230
6231 \newcommand\mhtikzinput [2] [] {%
6232     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
6233     \stex_in_repository:nn\Gin@mhrepos{
6234         \tikzinput [ #1 ] {\mhp{##1}{#2}}
6235     }
6236 }
6237 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
6238 </stex>

```

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

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.

```
6239 \*cls)
6240 \@@=document_structure)
6241 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6242 \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`

```
6243 \keys_define:nn{ document-structure / pkg }{
6244   class      .str_set_x:N = \c_document_structure_class_str,
6245   minimal    .bool_set:N = \c_document_structure_minimal_bool,
6246   report     .code:n      = {
6247     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6248     \str_set:Nn \c_document_structure_class_str {report}
6249   },
6250   book       .code:n      = {
6251     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6252     \str_set:Nn \c_document_structure_class_str {book}
6253   },
6254   bookpart   .code:n      = {
6255     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6256     \str_set:Nn \c_document_structure_class_str {book}
6257     \str_set:Nn \c_document_structure_topsect_str {chapter}
6258   },
```

```

6259 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6260 unknown     .code:n      = {
6261   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6262 }
6263 }
6264 \ProcessKeysOptions{ document-structure / pkg }
6265 \str_if_empty:NT \c_document_structure_class_str {
6266   \str_set:Nn \c_document_structure_class_str {article}
6267 }
6268 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6269   {\c_document_structure_class_str}
6270

```

37.3 Beefing up the document environment

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

```

6271 \RequirePackage{document-structure}
6272 \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

```

6273 \keys_define:nn { document-structure / document }{
6274   id .str_set_x:N = \c_document_structure_document_id_str
6275 }
6276 \let\__document_structure_orig_document=\document
6277 \renewcommand{\document}[1][]{
6278   \keys_set:nn{ document-structure / document }{ #1 }
6279   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6280   \__document_structure_orig_document
6281 }

```

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

```

6282 }
6283 \</cls>

```

37.4 Implementation: document-structure Package

```

6284 \<*package>
6285 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6286 \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

```

6287
6288 \keys_define:nn{ document-structure / pkg }{
6289   class      .str_set_x:N = \c_document_structure_class_str,
6290   topsect     .str_set_x:N = \c_document_structure_topsect_str,
6291   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6292 }
6293 \ProcessKeysOptions{ document-structure / pkg }
6294 \str_if_empty:NT \c_document_structure_class_str {
6295   \str_set:Nn \c_document_structure_class_str {article}
6296 }
6297 \str_if_empty:NT \c_document_structure_topsect_str {
6298   \str_set:Nn \c_document_structure_topsect_str {section}
6299 }

```

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

```

6300 \RequirePackage{xspace}
6301 \RequirePackage{comment}
6302 \AddToHook{begindocument}{
6303   \ltx@ifpackageloaded{babel}{
6304     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6305     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6306       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6307     }
6308   }{}
6309 }

```

`\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.

```

6310 \int_new:N \l_document_structure_section_level_int
6311 \str_case:VnF \c_document_structure_topsect_str {
6312   {part}}{
6313     \int_set:Nn \l_document_structure_section_level_int {0}
6314   }
6315   {chapter}}{
6316     \int_set:Nn \l_document_structure_section_level_int {1}
6317   }
6318 }{
6319   \str_case:VnF \c_document_structure_class_str {
6320     {book}}{
6321       \int_set:Nn \l_document_structure_section_level_int {0}
6322     }
6323     {report}}{
6324       \int_set:Nn \l_document_structure_section_level_int {0}
6325     }
6326   }{
6327     \int_set:Nn \l_document_structure_section_level_int {2}
6328   }
6329 }

```

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

```
6330 \def\current@section@level{document}%
6331 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6332 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6333 \cs_new_protected:Npn \skipomgroup {
6334   \ifcase\l_document_structure_section_level_int
6335   \or\stepcounter{part}
6336   \or\stepcounter{chapter}
6337   \or\stepcounter{section}
6338   \or\stepcounter{subsection}
6339   \or\stepcounter{subsubsection}
6340   \or\stepcounter{paragraph}
6341   \or\stepcounter{subparagraph}
6342   \fi
6343 }
```

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

`blindfragment`

```
6344 \newcommand\at@begin@blindomgroup[1]{%
6345   \newenvironment{blindfragment}
6346   {
6347     \int_incr:N\l_document_structure_section_level_int
6348     \at@begin@blindomgroup\l_document_structure_section_level_int
6349   }{}}
```

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

```
6350 \newcommand\omgroup@nonum[2]{
6351   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6352   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6353 }
```

(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.

```
6354 \newcommand\omgroup@num[2]{
```

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

```

6355 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6356   \@nameuse{#1}{#2}
6357 }{
6358   \cs_if_exist:NTF\rdmeta@sectioning{
6359     \@nameuse{rdmeta@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6360   }{
6361     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6362   }
6363 }
6364 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6365 }

```

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

sfragment

```

6366 \keys_define:nn { document-structure / omgroup }{
6367   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6368   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6369   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6370   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6371   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6372   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6373   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6374   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6375   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6376   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6377 }
6378 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
6379   \str_clear:N \l__document_structure_omgroup_id_str
6380   \str_clear:N \l__document_structure_omgroup_date_str
6381   \clist_clear:N \l__document_structure_omgroup_creators_clist
6382   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6383   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6384   \tl_clear:N \l__document_structure_omgroup_type_tl
6385   \tl_clear:N \l__document_structure_omgroup_short_tl
6386   \tl_clear:N \l__document_structure_omgroup_display_tl
6387   \tl_clear:N \l__document_structure_omgroup_intro_tl
6388   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6389   \keys_set:nn { document-structure / omgroup } { #1 }
6390 }

```

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.

```

6391 \newif\if@mainmatter\@mainmattertrue
6392 \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.

```

6393 \keys_define:nn { document-structure / sectioning }{
6394   name .str_set_x:N = \l__document_structure_sect_name_str ,
6395   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6396   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6397   clear .default:n = {true} ,
6398   num .bool_set:N = \l__document_structure_sect_num_bool ,

```



```

6399   num      .default:n      = {true}
6400 }
6401 \cs_new_protected:Nn \__document_structure_sect_args:n {
6402   \str_clear:N \l__document_structure_sect_name_str
6403   \str_clear:N \l__document_structure_sect_ref_str
6404   \bool_set_false:N \l__document_structure_sect_clear_bool
6405   \bool_set_false:N \l__document_structure_sect_num_bool
6406   \keys_set:nn { document-structure / sectioning } { #1 }
6407 }
6408 \newcommand\omdoc@sectioning[3][]{
6409   \__document_structure_sect_args:n {#1}
6410   \let\omdoc@sect@name\l__document_structure_sect_name_str
6411   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6412   \if@mainmatter% numbering not overridden by frontmatter, etc.
6413     \bool_if:NTF \l__document_structure_sect_num_bool {
6414       \omgroup@num{#2}{#3}
6415     }{
6416       \omgroup@nonum{#2}{#3}
6417     }
6418     \def\current@section@level{\omdoc@sect@name}
6419   \else
6420     \omgroup@nonum{#2}{#3}
6421   \fi
6422 }% 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.

```

6423 \newcommand\omgroup@redefine@addtocontents[1]{%
6424   %\edef\__document_structureimport{#1}%
6425   %\@for\@I:=\__document_structureimport\do{%
6426     %\edef\@path{\csname module@\@I @path\endcsname}%
6427     %\@ifundefined{tf@toc}\relax%
6428     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6429   %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6430   %\def\addcontentsline##1##2##3{%
6431     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6432   %\else% hyperref.sty not loaded
6433   %\def\addcontentsline##1##2##3{%
6434     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6435   %\fi
6436 }% 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.

```

6437 \newenvironment{sfragment}[2][]{% keys, title
6438 {
6439   \__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.

```

6440   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6441     \omgroup@redefine@addtocontents{
6442       %\@ifundefined{module@id}\used@modules%

```

```

6443     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6444   }
6445 }

```

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

```

6446 \int_incr:N\l_document_structure_section_level_int
6447 \ifcase\l_document_structure_section_level_int
6448   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6449   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6450   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6451   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6452   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6453   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6454   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6455 \fi
6456 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6457 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6458   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6459 }
6460 }% for customization
6461 {}

```

and finally, we localize the sections

```

6462 \newcommand\omdoc@part@kw{Part}
6463 \newcommand\omdoc@chapter@kw{Chapter}
6464 \newcommand\omdoc@section@kw{Section}
6465 \newcommand\omdoc@subsection@kw{Subsection}
6466 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6467 \newcommand\omdoc@paragraph@kw{paragraph}
6468 \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`

```

6469 \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).

```

6470 \cs_if_exist:NTF\frontmatter{
6471   \let\__document_structure_orig_frontmatter\frontmatter
6472   \let\frontmatter\relax
6473 }{
6474   \tl_set:Nn\__document_structure_orig_frontmatter{
6475     \clearpage
6476     \@mainmatterfalse
6477     \pagenumbering{roman}

```

```

6478 }
6479 }
6480 \cs_if_exist:NTF\backmatter{
6481   \let\__document_structure_orig_backmatter\backmatter
6482   \let\backmatter\relax
6483 }{
6484   \tl_set:Nn\__document_structure_orig_backmatter{
6485     \clearpage
6486     \@mainmatterfalse
6487     \pagenumbering{roman}
6488   }
6489 }

```

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.

```

6490 \newenvironment{frontmatter}{
6491   \__document_structure_orig_frontmatter
6492 }{
6493   \cs_if_exist:NTF\mainmatter{
6494     \mainmatter
6495   }{
6496     \clearpage
6497     \@mainmattertrue
6498     \pagenumbering{arabic}
6499   }
6500 }

```

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

```

6501 \newenvironment{backmatter}{
6502   \__document_structure_orig_backmatter
6503 }{
6504   \cs_if_exist:NTF\mainmatter{
6505     \mainmatter
6506   }{
6507     \clearpage
6508     \@mainmattertrue
6509     \pagenumbering{arabic}
6510   }
6511 }

```

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

```

6512 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6513 \def \c__document_structure_document_str{document}
6514 \newcommand\afterprematurestop{}
6515 \def\prematurestop@endomgroup{
6516   \unless\ifx\@currenvir\c__document_structure_document_str
6517     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
6518     \expandafter\prematurestop@endomgroup

```

```

6519 \fi
6520 }
6521 \providecommand\prematurestop{
6522   \message{Stopping~sTeX~processing~prematurely}
6523   \prematurestop@endomgroup
6524   \afterprematurestop
6525   \end{document}
6526 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

6527 \RequirePackage{etoolbox}
6528 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6529 \newrobustcmd\useSGvar[1]{%
6530   \@ifundefined{sTeX@Gvar@#1}
6531   {\PackageError{document-structure}
6532     {The sTeX Global variable #1 is undefined}
6533     {set it with \protect\setSGvar}}
6534   \@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.

```

6535 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6536   \@ifundefined{sTeX@Gvar@#1}
6537   {\PackageError{document-structure}
6538     {The sTeX Global variable #1 is undefined}
6539     {set it with \protect\setSGvar}}
6540   {\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.

```
6541 \*cls)
6542 \@@=notesslides)
6543 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6544 \RequirePackage{13keys2e}
6545
6546 \keys_define:nn{notesslides / cls}{
6547   class .code:n = {
6548     \PassOptionsToClass{\CurrentOption}{document-structure}
6549     \str_if_eq:nnT{#1}{book}{
6550       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6551     }
6552     \str_if_eq:nnT{#1}{report}{
6553       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6554     }
6555   },
6556   notes .bool_set:N = \c__notesslides_notes_bool ,
6557   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6558   unknown .code:n = {
6559     \PassOptionsToClass{\CurrentOption}{document-structure}
6560     \PassOptionsToClass{\CurrentOption}{beamer}
6561     \PassOptionsToPackage{\CurrentOption}{notesslides}
6562   }
6563 }
6564 \ProcessKeysOptions{ notesslides / cls }
6565 \bool_if:NTF \c__notesslides_notes_bool {
6566   \PassOptionsToPackage{notes=true}{notesslides}
6567 }{
6568   \PassOptionsToPackage{notes=false}{notesslides}
6569 }
6570 \</cls)
```

now we do the same for the notesslides package.

```

6571 <*package>
6572 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6573 \RequirePackage{13keys2e}
6574
6575 \keys_define:nn{notesslides / pkg}{
6576   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6577   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6578   notes        .bool_set:N = \c_notesslides_notes_bool ,
6579   slides        .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6580   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6581   frameimages   .bool_set:N = \c_notesslides_frameimages_bool ,
6582   fiboxed        .bool_set:N = \c_notesslides_fiboxed_bool ,
6583   nopproblems    .bool_set:N = \c_notesslides_nopproblems_bool,
6584   unknown        .code:n      = {
6585     \PassOptionsToClass{\CurrentOption}{stex}
6586     \PassOptionsToClass{\CurrentOption}{tikzinput}
6587   }
6588 }
6589 \ProcessKeysOptions{ notesslides / pkg }
6590 \newif\ifnotes
6591 \bool_if:NTF \c_notesslides_notes_bool {
6592   \notesttrue
6593 }{
6594   \notesfalse
6595 }
6596

```

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

```

6597 \str_if_empty:NTF \c_notesslides_topsect_str {
6598   \str_set_eq:NN \__notesslidestopsect \c_notesslides_defaulttopsec_str
6599 }{
6600   \str_set_eq:NN \__notesslidestopsect \c_notesslides_topsect_str
6601 }
6602 </package>

```

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

```

6603 <*cls>
6604 \bool_if:NTF \c_notesslides_notes_bool {
6605   \LoadClass{document-structure}
6606 }{
6607   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6608   \newcounter{Item}
6609   \newcounter{paragraph}
6610   \newcounter{subparagraph}
6611   \newcounter{Hfootnote}
6612   \RequirePackage{document-structure}
6613 }

```

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

```

6614 \RequirePackage{notesslides}
6615 </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.

```

6616 <*package>
6617 \bool_if:NT \c__notesslides_notes_bool {
6618   \RequirePackage{a4wide}
6619   \RequirePackage{marginnote}
6620   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6621   \RequirePackage{mdframed}
6622   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6623   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6624 }
6625 \RequirePackage{stex-tikzinput}
6626 \RequirePackage{etoolbox}
6627 \RequirePackage{amssymb}
6628 \RequirePackage{amsmath}
6629 \RequirePackage{comment}
6630 \RequirePackage{textcomp}
6631 \RequirePackage{url}
6632 \RequirePackage{graphicx}
6633 \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

```

6634 \bool_if:NT \c__notesslides_notes_bool {
6635   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6636 }
6637
6638
6639 \NewDocumentCommand \libusetheme {0{} m} {
6640   \bool_if:NTF \c__notesslides_notes_bool {
6641     \libusepackage[#1]{beamernotestheme#2}
6642   }{
6643     \libusepackage[#1]{beamertheme#2}
6644   }
6645 }

```

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

```

6646 \newcounter{slide}
6647 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6648 \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.

```

6649 \bool_if:NTF \c__notesslides_notes_bool {
6650   \renewenvironment{note}{\ignorespaces}{}
6651 }{
6652   \excludecomment{note}
6653 }

```

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.

```

6654 \bool_if:NT \c__notesslides_notes_bool {
6655   \newlength{\slideframewidth}
6656   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6657 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6658   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6659     \bool_set_true:N #1
6660   }{
6661     \bool_set_false:N #1
6662   }
6663 }
6664 \keys_define:nn{notesslides / frame}{
6665   label .str_set_x:N = \l__notesslides_frame_label_str,
6666   allowframebreaks .code:n = {
6667     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6668   },
6669   allowdisplaybreaks .code:n = {
6670     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6671   },
6672   fragile .code:n = {
6673     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6674   },
6675   shrink .code:n = {
6676     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6677   },
6678   squeeze .code:n = {
6679     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6680   },
6681   t .code:n = {
6682     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6683   },
6684 }
6685 \cs_new_protected:Nn \__notesslides_frame_args:n {
6686   \str_clear:N \l__notesslides_frame_label_str
6687   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6688   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6689   \bool_set_true:N \l__notesslides_frame_fragile_bool
6690   \bool_set_true:N \l__notesslides_frame_shrink_bool
6691   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6692   \bool_set_true:N \l__notesslides_frame_t_bool

```



```

6693     \keys_set:nn { notesslides / frame }{ #1 }
6694 }

```

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

```

6695 \renewenvironment{frame}[1][]{
6696   \__notesslides_frame_args:n{#1}
6697   \sffamily
6698   \stepcounter{slide}
6699   \def\@currentlabel{\theslide}
6700   \str_if_empty:NF \l__notesslides_frame_label_str {
6701     \label{\l__notesslides_frame_label_str}
6702   }

```

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

```

6703   \def\itemize@level{outer}
6704   \def\itemize@outer{outer}
6705   \def\itemize@inner{inner}
6706   \renewcommand\newpage{\addtocounter{framenumber}{1}}
6707   \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6708   \renewenvironment{itemize}{
6709     \ifx\itemize@level\itemize@outer
6710       \def\itemize@label{$\rhd$}
6711     \fi
6712     \ifx\itemize@level\itemize@inner
6713       \def\itemize@label{$\scriptstyle\rhd$}
6714     \fi
6715     \begin{list}
6716       {\itemize@label}
6717       {\setlength{\labelsep}{.3em}
6718        \setlength{\labelwidth}{.5em}
6719        \setlength{\leftmargin}{1.5em}
6720       }
6721     \edef\itemize@level{\itemize@inner}
6722   }{
6723     \end{list}
6724   }

```

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

```

6725   \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6726   }{
6727     \medskip\miko@slidelabel\end{mdframed}
6728   }

```

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

`\frametitle`

```

6729   \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6730 }

```

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

EdN:15

`\pause`

```

15
6731 \bool_if:NT \c__notesslides_notes_bool {
6732   \newcommand\pause{}
6733 }

```

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

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

nparagraph

```
6734 \bool_if:NTF \c__notesslides_notes_bool {
6735   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}
6736 }{
6737   \excludecomment{nparagraph}
6738 }
```

nfragment

```
6739 \bool_if:NTF \c__notesslides_notes_bool {
6740   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}
6741 }{
6742   \excludecomment{nfragment}
6743 }
```

ndefinition

```
6744 \bool_if:NTF \c__notesslides_notes_bool {
6745   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}
6746 }{
6747   \excludecomment{ndefinition}
6748 }
```

nassertion

```
6749 \bool_if:NTF \c__notesslides_notes_bool {
6750   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}
6751 }{
6752   \excludecomment{nassertion}
6753 }
```

nsproof

```
6754 \bool_if:NTF \c__notesslides_notes_bool {
6755   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}
6756 }{
6757   \excludecomment{nsproof}
6758 }
```

nexample

```
6759 \bool_if:NTF \c__notesslides_notes_bool {
6760   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}
6761 }{
6762   \excludecomment{nexample}
6763 }
```

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

```
6764 \def\inputref@preskip{\smallskip}
6765 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6766 \let\orig@inputref\inputref
6767 \def\inputref{\@ifstar\ninputref\orig@inputref}
6768 \newcommand\ninputref[2][] {
6769   \bool_if:NT \c__notesslides_notes_bool {
6770     \orig@inputref[#1]{#2}
6771   }
6772 }
```

(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>}`.

```
6773 \newlength{\slidelogoheight}
6774
6775 \bool_if:NTF \c__notesslides_notes_bool {
6776   \setlength{\slidelogoheight}{.4cm}
6777 }{
6778   \setlength{\slidelogoheight}{1cm}
6779 }
6780 \newsavebox{\slidelogo}
6781 \sbox{\slidelogo}{\text{\TeX}}
6782 \newrobustcmd{\setslidelogo}[1]{\def\slidelogo{#1}}
6783 \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6784 }
```

(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.

```
6785 \def\source{Michael Kohlhase}% customize locally
6786 \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.

```
6787 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6788 \newsavebox{\cclogo}
6789 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6790 \newif\ifcchref\cchreffalse
6791 \AtBeginDocument{
6792   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6793 }
6794 \def\licensing{
6795   \ifcchref
```

```

6796     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6797 \else
6798     {\usebox{\cclogo}}
6799 \fi
6800 }
6801 \newrobustcmd{\setlicensing}[2][]{
6802     \def\@url{#1}
6803     \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6804     \ifx\@url\@empty
6805         \def\licensing{\usebox{\cclogo}}
6806     \else
6807         \def\licensing{
6808             \ifcchref
6809             \href{#1}{\usebox{\cclogo}}
6810             \else
6811             {\usebox{\cclogo}}
6812             \fi
6813         }
6814     \fi
6815 }

```

(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

```

6816 \newrobustcmd\miko@slidelabel{
6817     \vbox to \slidelogoheight{
6818         \vss\hbox to \slidewidth
6819         {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6820     }
6821 }

```

(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.

```

6822 \def\Gin@mhrepos{}
6823 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6824 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6825 \newrobustcmd\frameimage[2][]{
6826     \stepcounter{slide}
6827     \bool_if:NT \c__notesslides_frameimages_bool {
6828         \def\Gin@ewidth{}\setkeys{Gin}{#1}
6829         \bool_if:NF \c__notesslides_notes_bool { \vfill }
6830         \begin{center}
6831             \bool_if:NTF \c__notesslides_fiboxed_bool {
6832                 \fbox{
6833                     \ifx\Gin@ewidth\@empty
6834                     \ifx\Gin@mhrepos\@empty
6835                         \mhgraphics[width=\slidewidth,#1]{#2}
6836                     \else

```

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

```

6837         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6838     \fi
6839 \else% Gin@ewidth empty
6840     \ifx\Gin@mhrepos\@empty
6841         \mhgraphics[#1]{#2}
6842     \else
6843         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6844     \fi
6845 \fi% Gin@ewidth empty
6846 }
6847 }{
6848     \ifx\Gin@ewidth\@empty
6849     \ifx\Gin@mhrepos\@empty
6850         \mhgraphics[width=\slidewidth,#1]{#2}
6851     \else
6852         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6853     \fi
6854     \ifx\Gin@mhrepos\@empty
6855         \mhgraphics[#1]{#2}
6856     \else
6857         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6858     \fi
6859 \fi% Gin@ewidth empty
6860 }
6861 \end{center}
6862 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6863 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6864 }
6865 } % 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.

```

6866 \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

```

6867 \AddToHook{begindocument}{
6868     \definecolor{green}{rgb}{0,.5,0}
6869     \definecolor{purple}{cmyk}{.3,1,0,.17}
6870 }

```

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`.

```

6871 % \def\STpresent#1{\textcolor{blue}{#1}}
6872 \def\defemph#1{\textcolor{magenta}{#1}}
6873 \def\symrefemph#1{\textcolor{cyan}{#1}}
6874 \def\compemph#1{\textcolor{blue}{#1}}
6875 \def\titleemph#1{\textcolor{blue}{#1}}
6876 \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.

```

6877 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6878 \def\smalltextwarning{
6879   \pgfuseimage{miko@small@dbend}
6880   \xspace
6881 }
6882 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6883 \newrobustcmd\textwarning{
6884   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6885   \xspace
6886 }
6887 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6888 \newrobustcmd\bigtextwarning{
6889   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6890   \xspace
6891 }

```

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

```

6892 \newrobustcmd\putgraphicsat[3]{
6893   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6894 }
6895 \newrobustcmd\putat[2]{
6896   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6897 }

```

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`.

```

6898 \bool_if:NT \c__notesslides_sectocframes_bool {
6899   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6900     \newcounter{chapter}\counterwithin*{section}{chapter}
6901   }{
6902     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6903       \newcounter{chapter}\counterwithin*{section}{chapter}
6904     }
6905   }
6906 }

```

`\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
6907 \def\part@prefix{}
6908 \@ifpackageloaded{document-structure}{}{
6909   \str_case:VnF \__notesslidesstopsect {
6910     {part}{
6911       \int_set:Nn \l_document_structure_section_level_int {0}
6912       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6913     \def\part@prefix{\arabic{chapter}.}
6914   }
6915   {chapter}{
6916     \int_set:Nn \l_document_structure_section_level_int {1}
6917     \def\thesection{\arabic{chapter}.\arabic{section}}
6918     \def\part@prefix{\arabic{chapter}.}
6919   }
6920 }{
6921   \int_set:Nn \l_document_structure_section_level_int {2}
6922   \def\part@prefix{}
6923 }
6924 }
6925
6926 \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

```

6927 \renewenvironment{sfragment}[2][]{
6928   \_document_structure_omgroup_args:n { #1 }
6929   \int_incr:N \l_document_structure_section_level_int
6930   \bool_if:NT \c__notesslides_sectocframes_bool {
6931     \stepcounter{slide}
6932     \begin{frame}[noframenumbering]
6933     \vfill\Large\centering
6934     \red{
6935       \ifcase\l_document_structure_section_level_int\or
6936         \stepcounter{part}
6937         \def\_notesslideslabel{\omdoc@part@kw~\Roman{part}}
6938         \def\currentsectionlevel{\omdoc@part@kw}
6939       \or
6940         \stepcounter{chapter}
6941         \def\_notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6942         \def\currentsectionlevel{\omdoc@chapter@kw}
6943       \or
6944         \stepcounter{section}
6945         \def\_notesslideslabel{\part@prefix\arabic{section}}
6946         \def\currentsectionlevel{\omdoc@section@kw}
6947       \or
6948         \stepcounter{subsection}
6949         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6950         \def\currentsectionlevel{\omdoc@subsection@kw}
6951       \or
6952         \stepcounter{subsubsection}
6953         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6954         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6955       \or
6956         \stepcounter{paragraph}
6957         \def\_notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6958         \def\currentsectionlevel{\omdoc@paragraph@kw}
6959       \else
6960         \def\_notesslideslabel{}

```

```

6961         \def\currentsectionlevel{\omdoc@paragraph@kw}
6962         \fi% end ifcase
6963         \_notesslideslabel%\sref@label@id\_notesslideslabel
6964         \quad #2%
6965     }%
6966     \vfill%
6967     \end{frame}%
6968 }
6969 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6970     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6971 }
6972 }{}
6973 }

```

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

```

6974 \def\inserttheorembodyfont{\normalfont}
6975 %\bool_if:NF \c__notesslides_notes_bool {
6976 % \defbeamertemplate{theorem begin}{miko}
6977 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6978 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6979 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6980 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6981 % \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.

```

6982 % \expandafter\def\csname Parent2\endcsname{}
6983 %}
6984
6985 \AddToHook{begindocument}{% this does not work for some reason
6986     \setbeamertemplate{theorems}[ams style]
6987 }
6988 \bool_if:NT \c__notesslides_notes_bool {
6989     \renewenvironment{columns}[1][{}]{%
6990         \par\noindent%
6991         \begin{minipage}%
6992             \slidewidth\centering\leavevmode%
6993     }{}%
6994     \end{minipage}\par\noindent%
6995 }%
6996 \newsavebox\columnbox%
6997 \renewenvironment<>{column}[2][{}]{%
6998     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6999 }{}%
7000     \end{minipage}\end{lrbox}\usebox\columnbox%
7001 }%
7002 }
7003 \bool_if:NTF \c__notesslides_noproblems_bool {
7004     \newenvironment{problems}{}{}
7005 }{
7006     \excludecomment{problems}
7007 }

```


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.

```

7008 \gdef\printexcursions{}
7009 \newcommand\excursionref[2]{% label, text
7010   \bool_if:NT \c__notesslides_notes_bool {
7011     \begin{sparagraph}[title=Excursion]
7012       #2 \sref[fallback=the appendix]{#1}.
7013     \end{sparagraph}
7014   }
7015 }
7016 \newcommand\activate@excursion[2][]{
7017   \gappto\printexcursions{\inputref{#1}{#2}}
7018 }
7019 \newcommand\excursion[4][]{% repos, label, path, text
7020   \bool_if:NT \c__notesslides_notes_bool {
7021     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
7022   }
7023 }

```

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

`\excursiongroup`

```

7024 \keys_define:nn{notesslides / excursiongroup }{
7025   id          .str_set_x:N = \l__notesslides_excursion_id_str,
7026   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
7027   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
7028 }
7029 \cs_new_protected:Nn \__notesslides_excursion_args:n {
7030   \tl_clear:N \l__notesslides_excursion_intro_tl
7031   \str_clear:N \l__notesslides_excursion_id_str
7032   \str_clear:N \l__notesslides_excursion_mhrepos_str
7033   \keys_set:nn {notesslides / excursiongroup }{ #1 }
7034 }
7035 \newcommand\excursiongroup[1][]{
7036   \__notesslides_excursion_args:n{ #1 }
7037   \ifdefempty\printexcursions{}% only if there are excursions
7038   {\begin{note}
7039     \begin{sfragment}[#1]{Excursions}%
7040     \ifdefempty\l__notesslides_excursion_intro_tl{\
7041       \inputref[\l__notesslides_excursion_mhrepos_str]{
7042         \l__notesslides_excursion_intro_tl
7043       }
7044     }
7045     \printexcursions%
7046     \end{sfragment}
7047   \end{note}}
7048 }
7049 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
7050 \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.

```
7051 <*package>
7052 <@@=problems>
7053 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
7054 \RequirePackage{l3keys2e,stex}
7055
7056 \keys_define:nn { problem / pkg }{
7057   notes      .default:n    = { true },
7058   notes      .bool_set:N   = \c__problems_notes_bool,
7059   gnotes     .default:n    = { true },
7060   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
7061   hints      .default:n    = { true },
7062   hints      .bool_set:N   = \c__problems_hints_bool,
7063   solutions  .default:n    = { true },
7064   solutions  .bool_set:N   = \c__problems_solutions_bool,
7065   pts        .default:n    = { true },
7066   pts        .bool_set:N   = \c__problems_pts_bool,
7067   min        .default:n    = { true },
7068   min        .bool_set:N   = \c__problems_min_bool,
7069   boxed      .default:n    = { true },
7070   boxed      .bool_set:N   = \c__problems_boxed_bool,
7071   unknown    .code:n       = {}
7072 }
7073 \newif\ifsolutions
7074
7075 \ProcessKeysOptions{ problem / pkg }
7076 \bool_if:NTF \c__problems_solutions_bool {
7077   \solutionstrue
7078 }{
7079   \solutionsfalse
7080 }
```

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

```
7081 \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.

```
7082 \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.

```
7083 \def\prob@problem@kw{Problem}
7084 \def\prob@solution@kw{Solution}
7085 \def\prob@hint@kw{Hint}
7086 \def\prob@note@kw{Note}
7087 \def\prob@gnote@kw{Grading}
7088 \def\prob@pt@kw{pt}
7089 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
7090 \AddToHook{begindocument}{
7091   \ltx@ifpackageloaded{babel}{
7092     \makeatletter
7093     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7094     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7095       \input{problem-ngerman.ldf}
7096     }
7097     \clist_if_in:NnT \l_tmpa_clist {finnish}{
7098       \input{problem-finnish.ldf}
7099     }
7100     \clist_if_in:NnT \l_tmpa_clist {french}{
7101       \input{problem-french.ldf}
7102     }
7103     \clist_if_in:NnT \l_tmpa_clist {russian}{
7104       \input{problem-russian.ldf}
7105     }
7106     \makeatother
7107   }{ }
7108 }
```

39.2 Problems and Solutions

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

```
7109 \keys_define:nn{ problem / problem }{
7110   id      .str_set:x:N = \l__problems_prob_id_str,
7111   pts     .tl_set:N    = \l__problems_prob_pts_tl,
7112   min     .tl_set:N    = \l__problems_prob_min_tl,
7113   title   .tl_set:N    = \l__problems_prob_title_tl,
7114   type    .tl_set:N    = \l__problems_prob_type_tl,
7115   refnum  .int_set:N   = \l__problems_prob_refnum_int
7116 }
7117 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

7118 \str_clear:N \l__problems_prob_id_str
7119 \tl_clear:N \l__problems_prob_pts_tl
7120 \tl_clear:N \l__problems_prob_min_tl
7121 \tl_clear:N \l__problems_prob_title_tl
7122 \tl_clear:N \l__problems_prob_type_tl
7123 \int_zero_new:N \l__problems_prob_refnum_int
7124 \keys_set:nn { problem / problem }{ #1 }
7125 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
7126   \let\l__problems_prob_refnum_int\undefined
7127 }
7128 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

7129 \newcounter{problem}
7130 \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.

```

7131 \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

```

7132 \newcommand\prob@number{
7133   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
7134     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
7135   }{
7136     \int_if_exist:NTF \l__problems_prob_refnum_int {
7137       \prob@label{\int_use:N \l__problems_prob_refnum_int }
7138     }{
7139       \prob@label\theproblem
7140     }
7141   }
7142 }

```

(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.

```

7143 \newcommand\prob@title[3]{%
7144   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
7145     #2 \l__problems_inclprob_title_tl #3
7146   }{
7147     \tl_if_exist:NTF \l__problems_prob_title_tl {
7148       #2 \l__problems_prob_title_tl #3
7149     }{
7150       #1
7151     }
7152   }
7153 }

```

(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.

```

7154 \def\prob@heading{
7155   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
7156   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
7157 }

```

(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`

```

7158 \newenvironment{sproblem}[1][{}]{
7159   \__problems_prob_args:n{#1}%\sref@target%
7160   \@in@omtexttrue% we are in a statement (for inline definitions)
7161   \stepcounter{problem}\record@problem
7162   \def\current@section@level{\prob@problem@kw}
7163   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
7164     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
7165   }{
7166     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
7167   }
7168   \str_if_exist:NTF \l__problems_inclprob_id_str {
7169     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
7170   }{
7171     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
7172   }
7173
7174
7175   \clist_set:No \l_tmpa_clist \sproblemtype
7176   \tl_clear:N \l_tmpa_tl
7177   \clist_map_inline:Nn \l_tmpa_clist {
7178     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
7179       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
7180     }
7181   }
7182   \tl_if_empty:NTF \l_tmpa_tl {
7183     \__problems_sproblem_start:
7184   }{
7185     \l_tmpa_tl
7186   }
7187   \stex_ref_new_doc_target:n \sproblemid
7188 }{
7189   \clist_set:No \l_tmpa_clist \sproblemtype
7190   \tl_clear:N \l_tmpa_tl
7191   \clist_map_inline:Nn \l_tmpa_clist {
7192     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
7193       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
7194     }

```

```

7195 }
7196 \tl_if_empty:NTF \l_tmpa_tl {
7197   \__problems_sproblem_end:
7198 }{
7199   \l_tmpa_tl
7200 }
7201
7202
7203 \smallskip
7204 }
7205
7206
7207 \cs_new_protected:Nn \__problems_sproblem_start: {
7208   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7209 }
7210 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7211
7212 \newcommand\stexpatchproblem[3][] {
7213   \str_set:Nx \l_tmpa_str{ #1 }
7214   \str_if_empty:NTF \l_tmpa_str {
7215     \tl_set:Nn \__problems_sproblem_start: { #2 }
7216     \tl_set:Nn \__problems_sproblem_end: { #3 }
7217   }{
7218     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7219     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7220   }
7221 }
7222
7223
7224 \bool_if:NT \c__problems_boxed_bool {
7225   \surroundwithmdframed{problem}
7226 }

```

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

```

7227 \def\record@problem{
7228   \protected@write\@auxout{}
7229   {
7230     \string\@problem{\prob@number}
7231     {
7232       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7233         \l__problems_inclprob_pts_tl
7234       }{
7235         \l__problems_prob_pts_tl
7236       }
7237     }%
7238     {
7239       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7240         \l__problems_inclprob_min_tl
7241       }{
7242         \l__problems_prob_min_tl
7243       }
7244     }
7245   }
7246 }

```

(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).

```
7247 \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.

```
7248 \keys_define:nn { problem / solution }{
7249   id                .str_set_x:N = \l__problems_solution_id_str ,
7250   for               .tl_set:N    = \l__problems_solution_for_tl ,
7251   height            .dim_set:N   = \l__problems_solution_height_dim ,
7252   creators           .clist_set:N = \l__problems_solution_creators_clist ,
7253   contributors       .clist_set:N = \l__problems_solution_contributors_clist ,
7254   srccite           .tl_set:N    = \l__problems_solution_srccite_tl
7255 }
7256 \cs_new_protected:Nn \__problems_solution_args:n {
7257   \str_clear:N \l__problems_solution_id_str
7258   \tl_clear:N \l__problems_solution_for_tl
7259   \tl_clear:N \l__problems_solution_srccite_tl
7260   \clist_clear:N \l__problems_solution_creators_clist
7261   \clist_clear:N \l__problems_solution_contributors_clist
7262   \dim_zero:N \l__problems_solution_height_dim
7263   \keys_set:nn { problem / solution }{ #1 }
7264 }
```

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

```
7265 \newcommand\@startsolution[1][{}]{
7266   \__problems_solution_args:n { #1 }
7267   \@in@omtexttrue% we are in a statement.
7268   \bool_if:NF \c__problems_boxed_bool { \hrule }
7269   \smallskip\noindent
7270   {\textbf\prob@solution@kw : \enspace}
7271   \begin{small}
7272   \def\current@section@level{\prob@solution@kw}
7273   \ignorespacesandpars
7274 }
```

\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.

```
7275 \newcommand\startsolutions{
7276   \specialcomment{solution}{\@startsolution}{
7277     \bool_if:NF \c__problems_boxed_bool {
7278       \hrule\medskip
7279     }
7280     \end{small}%
7281   }
7282   \bool_if:NT \c__problems_boxed_bool {
7283     \surroundwithmdframed{solution}
7284   }
7285 }
```

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

\stopsolutions

```
7286 \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.

```
7287 \ifsolutions
7288 \startsolutions
7289 \else
7290 \stopsolutions
7291 \fi
```

exnote

```
7292 \bool_if:NTF \c__problems_notes_bool {
7293   \newenvironment{exnote}[1][]{
7294     \par\smallskip\hrule\smallskip
7295     \noindent\textbf{\prob@note@kw : }\small
7296   }{
7297     \smallskip\hrule
7298   }
7299   }{
7300     \excludecomment{exnote}
7301   }
```

hint

```
7302 \bool_if:NTF \c__problems_notes_bool {
7303   \newenvironment{hint}[1][]{
7304     \par\smallskip\hrule\smallskip
7305     \noindent\textbf{\prob@hint@kw :~ }\small
7306   }{
7307     \smallskip\hrule
7308   }
7309   \newenvironment{exhint}[1][]{
7310     \par\smallskip\hrule\smallskip
7311     \noindent\textbf{\prob@hint@kw :~ }\small
7312   }{
7313     \smallskip\hrule
7314   }
7315   }{
7316     \excludecomment{hint}
7317     \excludecomment{exhint}
7318   }
```

gnote

```
7319 \bool_if:NTF \c__problems_notes_bool {
7320   \newenvironment{gnote}[1][]{
7321     \par\smallskip\hrule\smallskip
7322     \noindent\textbf{\prob@gnote@kw : }\small
7323   }{
7324     \smallskip\hrule
7325   }
7326   }{
7327     \excludecomment{gnote}
7328   }
```


39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7329 \newenvironment{mcb}{
7330   \begin{enumerate}
7331 }{
7332   \end{enumerate}
7333 }
```

we define the keys for the mcc macro

```

7344 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7345   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7346     \bool_set_true:N #1
7347   }{
7348     \bool_set_false:N #1
7349   }
7350 }
7351 \keys_define:nn { problem / mcc }{
7352   id          .str_set:x:N = \l__problems_mcc_id_str ,
7353   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7354   T           .default:n    = { true } ,
7355   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7356   F           .default:n    = { true } ,
7357   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7358   Ttext       .code:n       = {
7359     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7360   } ,
7361   Ftext       .code:n       = {
7362     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7363   }
7364 }
7365 \cs_new_protected:Nn \l__problems_mcc_args:n {
7366   \str_clear:N \l__problems_mcc_id_str
7367   \tl_clear:N \l__problems_mcc_feedback_tl
7368   \bool_set_true:N \l__problems_mcc_t_bool
7369   \bool_set_true:N \l__problems_mcc_f_bool
7370   \bool_set_true:N \l__problems_mcc_Ttext_bool
7371   \bool_set_false:N \l__problems_mcc_Ftext_bool
7372   \keys_set:nn { problem / mcc }{ #1 }
7373 }
```

\mcc

```

7364 \newcommand\mcc[2][]{
7365   \l__problems_mcc_args:n{ #1 }
7366   \item #2
7367   \ifsolutions
7368     \l
7369     \bool_if:NT \l__problems_mcc_t_bool {
7370       % TODO!
7371       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
7372     }
7373     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

7374      % TODO!
7375      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7376    }
7377    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7378      !
7379    }{
7380      \l__problems_mcc_feedback_tl
7381    }
7382    \fi
7383  } %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.

```

7384
7385 \keys_define:nn{ problem / inclproblem }{
7386   id      .str_set:N = \l__problems_inclprob_id_str,
7387   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7388   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7389   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7390   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7391   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7392   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
7393 }
7394 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7395   \str_clear:N \l__problems_prob_id_str
7396   \tl_clear:N \l__problems_inclprob_pts_tl
7397   \tl_clear:N \l__problems_inclprob_min_tl
7398   \tl_clear:N \l__problems_inclprob_title_tl
7399   \tl_clear:N \l__problems_inclprob_type_tl
7400   \int_zero_new:N \l__problems_inclprob_refnum_int
7401   \str_clear:N \l__problems_inclprob_mhrepos_str
7402   \keys_set:nn { problem / inclproblem }{ #1 }
7403   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7404     \let\l__problems_inclprob_pts_tl\undefined
7405   }
7406   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7407     \let\l__problems_inclprob_min_tl\undefined
7408   }
7409   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7410     \let\l__problems_inclprob_title_tl\undefined
7411   }
7412   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7413     \let\l__problems_inclprob_type_tl\undefined
7414   }
7415   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7416     \let\l__problems_inclprob_refnum_int\undefined
7417   }
7418 }

```

```

7419
7420 \cs_new_protected:Nn \__problems_inclprob_clear: {
7421   \let\l__problems_inclprob_id_str\undefined
7422   \let\l__problems_inclprob_pts_tl\undefined
7423   \let\l__problems_inclprob_min_tl\undefined
7424   \let\l__problems_inclprob_title_tl\undefined
7425   \let\l__problems_inclprob_type_tl\undefined
7426   \let\l__problems_inclprob_refnum_int\undefined
7427   \let\l__problems_inclprob_mhrepos_str\undefined
7428 }
7429 \__problems_inclprob_clear:
7430
7431 \newcommand\includeproblem[2][ ]{
7432   \__problems_inclprob_args:n{ #1 }
7433   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7434     \input{#2}
7435   }{
7436     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7437       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7438     }
7439   }
7440   \__problems_inclprob_clear:
7441 }

```

(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.

```

7442 \AddToHook{enddocument}{
7443   \bool_if:NT \c__problems_pts_bool {
7444     \message{Total:~\arabic{pts}~points}
7445   }
7446   \bool_if:NT \c__problems_min_bool {
7447     \message{Total:~\arabic{min}~minutes}
7448   }
7449 }

```

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

```

7450 \def\pts#1{
7451   \bool_if:NT \c__problems_pts_bool {
7452     \marginpar{#1~\prob@pt@kw}
7453   }
7454 }
7455 \def\min#1{
7456   \bool_if:NT \c__problems_min_bool {
7457     \marginpar{#1~\prob@min@kw}
7458   }
7459 }

```

`\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.

```

7460 \newcounter{pts}
7461 \def\show@pts{
7462   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7463     \bool_if:NT \c__problems_pts_bool {
7464       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7465       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7466     }
7467   }{
7468     \tl_if_exist:NT \l__problems_prob_pts_tl {
7469       \bool_if:NT \c__problems_pts_bool {
7470         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7471         \addtocounter{pts}{\l__problems_prob_pts_tl}
7472       }
7473     }
7474   }
7475 }

```

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

`\show@min`

```

7476 \newcounter{min}
7477 \def\show@min{
7478   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7479     \bool_if:NT \c__problems_min_bool {
7480       \marginpar{\l__problems_inclprob_min_tl\ min}
7481       \addtocounter{min}{\l__problems_inclprob_min_tl}
7482     }
7483   }{
7484     \tl_if_exist:NT \l__problems_prob_min_tl {
7485       \bool_if:NT \c__problems_min_bool {
7486         \marginpar{\l__problems_prob_min_tl\ min}
7487         \addtocounter{min}{\l__problems_prob_min_tl}
7488       }
7489     }
7490   }
7491 }
7492 \</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.

```
7493 \@@=hwexam>
7494 \*cls>
7495 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7496 \RequirePackage{l3keys2e}
7497 \DeclareOption*{
7498   \PassOptionsToClass{\CurrentOption}{document-structure}
7499   \PassOptionsToPackage{\CurrentOption}{stex}
7500   \PassOptionsToPackage{\CurrentOption}{hwexam}
7501   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7502 }
7503 \ProcessOptions
```

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

```
7504 \LoadClass{document-structure}
7505 \RequirePackage{stex}
7506 \RequirePackage{hwexam}
7507 \RequirePackage{tikzinput}
7508 \RequirePackage{graphicx}
7509 \RequirePackage{a4wide}
7510 \RequirePackage{amssymb}
7511 \RequirePackage{amstext}
7512 \RequirePackage{amsmath}
```

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

```

7513 \newcommand\assig@default@type{\hwexam@assignment@kw}
7514 \def\document@hwexamtype{\assig@default@type}
7515 <@@=document_structure>
7516 \keys_define:nn { document-structure / document }{
7517 id .str_set_x:N = \c_document_structure_document_id_str,
7518 hwexamtype .tl_set:N = \document@hwexamtype
7519 }
7520 <@@=hwexam>
7521 </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.

```
7522 \*package>
7523 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7524 \RequirePackage{13keys2e}
7525
7526 \newif\iftest\testfalse
7527 \DeclareOption{test}{\testtrue}
7528 \newif\ifmultiple\multiplefalse
7529 \DeclareOption{multiple}{\multipletrue}
7530 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7531 \ProcessOptions
```

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

```
7532 \RequirePackage{keyval}[1997/11/10]
7533 \RequirePackage{problem}
```

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

```
7534 \newcommand\hwexam@assignment@kw{Assignment}
7535 \newcommand\hwexam@given@kw{Given}
7536 \newcommand\hwexam@due@kw{Due}
7537 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7538 blank~for~extra~space}
7539 \def\hwexam@minutes@kw{minutes}
7540 \newcommand\correction@probs@kw{prob.}
7541 \newcommand\correction@pts@kw{total}
7542 \newcommand\correction@reached@kw{reached}
7543 \newcommand\correction@sum@kw{Sum}
7544 \newcommand\correction@grade@kw{grade}
7545 \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

```

7546 \AddToHook{begindocument}{
7547 \ltx@ifpackageloaded{babel}{
7548 \makeatletter
7549 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7550 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7551 \input{hwexam-ngerman.ldf}
7552 }
7553 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7554 \input{hwexam-finnish.ldf}
7555 }
7556 \clist_if_in:NnT \l_tmpa_clist {french}{
7557 \input{hwexam-french.ldf}
7558 }
7559 \clist_if_in:NnT \l_tmpa_clist {russian}{
7560 \input{hwexam-russian.ldf}
7561 }
7562 \makeatother
7563 }{}
7564 }
7565

```

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.

```

7566 \newcounter{assignment}
7567 \numberproblemsin{assignment}
7568 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7569 \keys_define:nn { hwexam / assignment } {
7570 id .str_set:N = \l__hwexam_assign_id_str,
7571 number .int_set:N = \l__hwexam_assign_number_int,
7572 title .tl_set:N = \l__hwexam_assign_title_tl,
7573 type .tl_set:N = \l__hwexam_assign_type_tl,
7574 given .tl_set:N = \l__hwexam_assign_given_tl,
7575 due .tl_set:N = \l__hwexam_assign_due_tl,
7576 loadmodules .code:n = {
7577 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7578 }
7579 }
7580 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7581 \str_clear:N \l__hwexam_assign_id_str
7582 \int_set:Nn \l__hwexam_assign_number_int {-1}
7583 \tl_clear:N \l__hwexam_assign_title_tl
7584 \tl_clear:N \l__hwexam_assign_type_tl
7585 \tl_clear:N \l__hwexam_assign_given_tl
7586 \tl_clear:N \l__hwexam_assign_due_tl
7587 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```



```

7588 \keys_set:nn { hwexam / assignment }{ #1 }
7589 }

```

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.

```

7590 \newcommand\given@due[2]{
7591 \bool_lazy_all:nF {
7592 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7593 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7594 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7595 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7596 }{ #1 }
7597
7598 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7599 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7600 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7601 }
7602 }{
7603 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7604 }
7605
7606 \bool_lazy_or:nnF {
7607 \bool_lazy_and_p:nn {
7608 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7609 }{
7610 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7611 }
7612 }{
7613 \bool_lazy_and_p:nn {
7614 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7615 }{
7616 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7617 }
7618 }{ ,~ }
7619
7620 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7621 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7622 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7623 }
7624 }{
7625 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7626 }
7627
7628 \bool_lazy_all:nF {
7629 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7630 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7631 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7632 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7633 }{ #2 }
7634 }

```

`\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.

```

7635 \newcommand\assignment@title[3]{
7636 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7637 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7638 #1
7639 }{
7640 #2\l__hwexam_assign_title_tl#3
7641 }
7642 }{
7643 #2\l__hwexam_inclasssign_title_tl#3
7644 }
7645 }

```

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

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

```

7646 \newcommand\assignment@number{
7647 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7648 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7649 \arabic{assignment}
7650 } {
7651 \int_use:N \l__hwexam_assign_number_int
7652 }
7653 }{
7654 \int_use:N \l__hwexam_inclasssign_number_int
7655 }
7656 }

```

(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.

```

7657 \newenvironment{assignment}[1][ ]{
7658 \__hwexam_assignment_args:n { #1 }
7659 %\sref@target
7660 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7661 \global\stepcounter{assignment}
7662 }{
7663 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7664 }
7665 \setcounter{problem}{0}
7666 \def\current@section@level{\document@hwexamtype}
7667 %\sref@label@id{\document@hwexamtype \thesection}
7668 \begin{@assignment}
7669 }{
7670 \end{@assignment}
7671 }

```

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

```

7672 \def\ass@title{
7673 \protect\document@hwexamtype~\arabic{assignment}
7674 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7675 }
7676 \ifmultiple
7677 \newenvironment{@assignment}{
7678 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7679 \begin{sfragment}[loadmodules]{\ass@title}
7680 }{
7681 \begin{sfragment}{\ass@title}
7682 }
7683 }{
7684 \end{sfragment}
7685 }

```

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

```

7686 \else
7687 \newenvironment{@assignment}{
7688 \begin{center}\bf
7689 \Large@title\strut\\
7690 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7691 \large\given@due{--\}{}\{;\}{}
7692 \end{center}
7693 }{}
7694 \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.

```

7695 \keys_define:nn { hwexam / inclassignment } {
7696 %id .str_set_x:N = \l__hwexam_assign_id_str,
7697 number .int_set:N = \l__hwexam_inclassign_number_int,
7698 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7699 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7700 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7701 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7702 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7703 }
7704 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7705 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7706 \tl_clear:N \l__hwexam_inclassign_title_tl
7707 \tl_clear:N \l__hwexam_inclassign_type_tl
7708 \tl_clear:N \l__hwexam_inclassign_given_tl
7709 \tl_clear:N \l__hwexam_inclassign_due_tl
7710 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7711 \keys_set:nn { hwexam / inclassignment }{ #1 }
7712 }
7713 \__hwexam_inclassignment_args:n {}
7714
7715 \newcommand\inputassignment[2][{}]{

```

```

7716 \_hwexam_inclassnment_args:n { #1 }
7717 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7718 \input{#2}
7719 }{
7720 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7721 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7722 }
7723 }
7724 \_hwexam_inclassnment_args:n {}
7725 }
7726 \newcommand\includeassignment[2][]{
7727 \newpage
7728 \inputassignment[#1]{#2}
7729 }

```

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

41.4 Typesetting Exams

\quizheading

```

7730 \ExplSyntaxOff
7731 \newcommand\quizheading[1]{%
7732 \def\@tas{#1}%
7733 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7734 \ifx\@tas\@empty\else%
7735 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7736 \fi%
7737 }
7738 \ExplSyntaxOn

```

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

\testheading

```

7739
7740 \def\hwexamheader{\input{hwexam-default.header}}
7741
7742 \def\hwexamminutes{
7743 \tl_if_empty:NTF \testheading@duration {
7744 {\testheading@min}~\hwexam@minutes@kw
7745 }{
7746 \testheading@duration
7747 }
7748 }
7749
7750 \keys_define:nn { hwexam / testheading } {
7751 min .tl_set:N = \testheading@min,
7752 duration .tl_set:N = \testheading@duration,
7753 reqpts .tl_set:N = \testheading@reqpts,
7754 tools .tl_set:N = \testheading@tools
7755 }
7756 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7757 \tl_clear:N \testheading@min
7758 \tl_clear:N \testheading@duration

```

```

7759 \tl_clear:N \testheading@reqpts
7760 \tl_clear:N \testheading@tools
7761 \keys_set:nn { hwexam / testheading }{ #1 }
7762 }
7763 \newenvironment{testheading}[1][]{
7764   \_hwexam_testheading_args:n{ #1 }
7765   \newcount\check@time\check@time=\testheading@min
7766   \advance\check@time by -\theassignment@totalmin
7767   \newif\if@bonuspoints
7768   \tl_if_empty:NTF \testheading@reqpts {
7769     \@bonuspointsfalse
7770   }{
7771     \newcount\bonus@pts
7772     \bonus@pts=\theassignment@totalpts
7773     \advance\bonus@pts by -\testheading@reqpts
7774     \edef\bonus@pts{\the\bonus@pts}
7775     \@bonuspointstrue
7776   }
7777   \edef\check@time{\the\check@time}
7778
7779   \makeatletter\hwexamheader\makeatother
7780 }{
7781   \newpage
7782 }

```

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

\testspace

```

7783 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7784 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7785 \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.

```

7786 <@=problems>
7787 \renewcommand\@problem[3]{
7788   \stepcounter{assignment@probs}
7789   \def\__problemspts{#2}
7790   \ifx\__problemspts\@empty\else
7791     \addtocounter{assignment@totalpts}{#2}
7792   \fi
7793   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7794   \xdef\correction@probs{\correction@probs & #1}%
7795   \xdef\correction@pts{\correction@pts & #2}
7796   \xdef\correction@reached{\correction@reached &}

```

```

7797 }
7798 <@@=hwexam>

```

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

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

```

7799 \newcounter{assignment@probs}
7800 \newcounter{assignment@totalpts}
7801 \newcounter{assignment@totalmin}
7802 \def\correction@probs{\correction@probs@kw}
7803 \def\correction@pts{\correction@pts@kw}
7804 \def\correction@reached{\correction@reached@kw}
7805 \stepcounter{assignment@probs}
7806 \newcommand\correction@table{
7807 \resizebox{\textwidth}{!}{%
7808 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7809 &\multicolumn{\theassignment@probs}{c|}|%|
7810 {\footnotesize\correction@forgrading@kw} &\\ \hline
7811 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7812 \correction@pts & \theassignment@totalpts & \\ \hline
7813 \correction@reached & & \[.7cm]\hline
7814 \end{tabular}}
7815 </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}

```