

The sTeX3 Package *

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

2022-03-10

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 2022-03-10)

Contents

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

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

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

22	problem.sty: An Infrastructure for formatting Problems	86
22.1	Introduction	86
22.2	The User Interface	86
22.2.1	Package Options	86
22.2.2	Problems and Solutions	87
22.2.3	Multiple Choice Blocks	88
22.2.4	Including Problems	88
22.2.5	Reporting Metadata	88
22.3	Limitations	88
23	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	90
23.1	Introduction	91
23.2	The User Interface	91
23.2.1	Package and Class Options	91
23.2.2	Assignments	91
23.2.3	Typesetting Exams	91
23.2.4	Including Assignments	92
23.3	Limitations	92
IV	Implementation	94
24	gTeX-Basics Implementation	95
24.1	The gTeXDocument Class	95
24.2	Preliminaries	95
24.3	Messages and logging	96
24.4	HTML Annotations	97
24.5	Babel Languages	100
24.6	Auxiliary Methods	101
25	gTeX-MathHub Implementation	102
25.1	Generic Path Handling	102
25.2	PWD and kpsewhich	104
25.3	File Hooks and Tracking	105
25.4	MathHub Repositories	106
25.5	Using Content in Archives	110
26	gTeX-References Implementation	115
26.1	Document URIs and URLs	115
26.2	Setting Reference Targets	117
26.3	Using References	119
27	gTeX-Modules Implementation	122
27.1	The smodule environment	126
27.2	Invoking modules	131
28	gTeX-Module Inheritance Implementation	133
28.1	SMS Mode	133
28.2	Inheritance	136

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

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

Part I

Manual



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



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

Chapter 1

What is sTeX?

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

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

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

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

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

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

Chapter 2

Quickstart

2.1 Setup

2.1.1 The sTeX IDE

TODO: VSCode Plugin

2.1.2 Manual Setup

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

- **The sTeX-Package** available [here](#).
sTeX is also available on CTAN and in TeXLive.
- To make sure that sTeX too knows where to find its archives, we need to set a global system variable `MATHHUB`, that points to your local `MathHub`-directory (see [section 3.2](#)).

- **The Mmt System** available [here](#)¹. We recommend following the setup routine documented [here](#).

Following the setup routine (Step 3) will entail designating a `MathHub`-directory on your local file system, where the MMT system will look for sTeX/MMT content archives.

- **sTeX Archives** If we only care about L^ATeX and generating pdfs, we do not technically need MMT at all; however, we still need the `MATHHUB` system variable to be set. Furthermore, MMT can make downloading content archives we might want to use significantly easier, since it makes sure that all dependencies of (often highly interrelated) sTeX archives are cloned as well.

Once set up, we can run `mmt` in a shell and download an archive along with all of its dependencies like this: `lmh install <name-of-repository>`, or a whole *group* of archives; for example, `lmh install smglom` will download all `smglom` archives.

- **RuSTeX** The MMT system will also set up RuSTeX for you, which is used to generate (semantically annotated) `xhtml` from tex sources. In lieu of using MMT, you can also download and use RuSTeX directly [here](#).

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

2.2 A First \LaTeX Document

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

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

```

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

```

Compiling this document with `pdflatex` should yield the output

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

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

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

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

Remark 2.2.1:

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

Let's investigate this document in detail now:

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

smodule First, we open a new *module* called `GeometricSeries`. This module is assigned a *globally unique* identifier (URI), which (depending on your pdf viewer) should pop up in a tooltip if you hover over the word **geometric series**.

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

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

The `\importmodule`-statements make all \LaTeX symbols and associated semantic macros (e.g. `\infinitesum`, `\realdive`, `\realpower`) in the desired module available. Additionally, they “export” these symbols to all further modules which include the *current* module – i.e. if in some future module we would put `\importmodule{GeometricSeries}`, we would also have `\infinitesum` etc. at our disposal.

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

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

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

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

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

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

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

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

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

2.2.1 OMDoc/xhtml Conversion

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

TODO VSCode Plugin

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

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

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

```
<OMBIND>
  <OMID name="...?series?infinitiesum"/>
  <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/smgglom/calculus
narration-base: http://mathhub.info/smgglom/calculus
dependencies: smglom/arithmetics,smglom/sets,smglom/topology,
              smglom/mv,smglom/linear-algebra,smglom/algebra
responsible: Michael.Kohlhase@FAU.de
title: Elementary Calculus
teaser: Terminology for the mathematical study of change.
description: desc.html
```

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

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

3.2.4 Using Files in \TeX Archives Directly

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

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

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

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

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

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

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

Remark 3.2.1:

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

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

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

3.3 Module, Symbol and Notation Declarations

3.3.1 The `smodule`-Environment

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1

Input:

```

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

```

Output:

Hello World

\stexpatchmodule

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

For example:

Example 2

Input:

```

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

```

Output:

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

3.3.2 Declaring New Symbols and Notations

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

`\symdecl`

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

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

\hookrightarrow `\symdecl` introduces a new OMDoc/MMT constant in the current module (=OMDoc/MMT theory). Correspondingly, they get assigned the URI \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.
- \rightarrow Correspondingly, the name “type” should be taken with a grain of salt, since
- \rightarrow OMDoc/MMT – being foundation-independent – does not a priori implement a fixed typing system.

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

Example 13

Input:

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

Output:

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

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

Example 14

Input:

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

Output:

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

3.3.5 Precedences and Automated Bracketing

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

Example 15

Input:

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

Output:

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

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

Example 16

Input:

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

Output:

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

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

Example 17

Input:

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

Output:

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

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

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

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

Example 18

Input:

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

Output:

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

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

`\infprec`
`\neginfprec`

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



More precisely, each notation takes

1. One *operator precedence* and

2. one *argument precedence* for each argument.

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

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

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

In the example above:



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

3.3.6 Variables

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

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

`\svar`

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

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

\vardef

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

Example 19

Input:

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

Output:

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

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

TODO: bind=forall/exists

3.3.7 Variable Sequences

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

\varseq

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

This is best shown by example:

Example 20

Input:

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

Output:

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

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

TODO: more notations for invoking sequences.

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

Example 21

Input:

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

Output:

$$a_1 + \dots + a_n$$

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

Example 22

Input:

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

Output:

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

We can also explicitly provide a “middle” segment to be used, like such:

Example 23

Input:

```
1 \varseq{seqa}[
2   name=a,
3   type=\Nat,
4   args=2,
5   mid={\comp{a}_{\varn}^1,\comp{a}_1^2,\ellipses,\comp{a}_1^{\varm}}
6 ]{1,1}{\varn,\varm}{\comp{a}_{#1}^{#2}}
7
8 \seqa! and \addition{\seqa}
```

Output:

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

3.4 Module Inheritance and Structures

3.4.1 Multilinguality and Translations

If we load the \TeX document class or package with the option `lang=<lang>`, \TeX will load the appropriate `babel` language for you – e.g. `lang=de` will load the `babel` language `ngerman`. Additionally, it makes \TeX aware of the current document being set in (in this example) *german*. This matters for reasons other than mere `babel`-purposes, though:

Every *module* is assigned a language. If no \TeX package option is set that allows for inferring a language, \TeX will check whether the current file name ends in e.g. `.en.tex` (or `.de.tex` or `.fr.tex`, or...) and set the language accordingly. Alternatively, a language can be explicitly assigned via `\begin{smodule}[lang=<language>]{Foo}`.

Technically, each `smodule`-environment induces *two* OMDoc/MMT theories:
 $\begin{array}{ll} \text{---M---} & \text{\begin{smodule}[lang=<lang>]{Foo} generates a theory some/namespace?Foo} \\ \text{---M---} & \text{that only contains the “formal” part of the module – i.e. exactly the content} \\ \text{---T---} & \text{that is exported when using \importmodule.} \\ \text{---T---} & \text{Additionally, MMT generates a language theory some/namespace/Foo?<lang> that} \\ & \text{includes some/namespace?Foo and contains all the other document content – vari-} \\ & \text{able declarations, includes for each \usemodule, etc.} \end{array}$

Notably, the language suffix in a filename is ignored for `\usemodule`, `\importmodule` and in generating/computing URIs for modules. This however allows for providing *translations* for modules between languages without needing to duplicate content:

If a module `Foo` exists in e.g. *english* in a file `Foo.en.tex`, we can provide a file `Foo.de.tex` right next to it, and write `\begin{smodule}[sig=en]{Foo}`. The `sig`-key then signifies, that the “signature” of the module is contained in the *english* version of the module, which is immediately imported from there, just like `\importmodule` would.

Additionally to translating the informal content of a module file to different languages, it also allows for customizing notations between languages. For example, the *least common multiple* of two numbers is often denoted as $\text{lcm}(a, b)$ in *english*, but is called *kleinstes gemeinsames Vielfaches* in *german* and consequently denoted as $\text{kgV}(a, b)$ there.

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

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

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

3.4.2 Simple Inheritance and Namespaces

`\importmodule`
`\usemodule`

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

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

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



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

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

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

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

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



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

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

The module `Foo` must either be declared in the



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

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

`\STEXexport`

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



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

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

3.4.3 The `mathstructure` Environment

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

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

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

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

How this works is again best demonstrated by example:

Example 24

Input:

```

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

Output:

A monoid is...

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

Example 25

Input:

```

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

Output:

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

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

Example 26

Input:

```

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

```

Output:

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

\instantiate

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

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

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

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

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

\varinstantiate

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

This allows us to do things like:

Example 27

Input:

```

1 \varinstantiate{varM}{\monoid}{M}
2
3 A \symname{monoid} is a structure
4 $\varM!:=\mathstrut{\varM{universe},\varM{op}!,\varM{unit}}{\varM{op}!:\funtype{\varM{universe},\varM{universe}}{\varM{universe}}}$
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: $a \cdot c \cdot c$

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 $\text{\texttt{sTeX}}$ Metatheory)

TODO: metatheory documentation

Chapter 4

Using \TeX Symbols

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

4.1 `\symref` and its variants

`\symref`
`\symname`

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

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

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

Example 31

Input:

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

Output:

A natural number is...

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

`\Symname`

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

Example 32

Input:

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

Output:

Natural numbers are...



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

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

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

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

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

4.2 Marking Up Text and On-the-Fly Notations

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

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

Example 33

Input:

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

Output:

The sum of n and m is...

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

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

\arg

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

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

Example 34

Input:

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

Output:

Addition is...

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

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

Example 35

Input:

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

Output:

adding k yields...

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

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

Example 36

Input:

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

Output:

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

4.3 Referencing Symbols and Statements

TODO: references documentation

Chapter 5

sTEX Statements

5.1 Definitions, Theorems, Examples, Paragraphs

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

The corresponding environments for that are:

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

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

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

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

Example 37

Input:

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

Output:

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

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

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

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

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

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

Example 38

Input:

```

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

Output:

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

Axiom 5.1.2 (Associativity). \circ is associative

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

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

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

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

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

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

5.2 Proofs

TODO

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

Chapter 6

Highlighting and Presentation Customizations

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

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

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

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

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

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

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

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

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

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

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

```

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

```

Or, if we want all **sdefinitions** to use a predefined **definition**-environment, we can do

```

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

```

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

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

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

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

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

```

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

```

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

Chapter 7

Additional Packages

TODO: tikzinput documentation

7.1 Modular Document Structuring

TODO: document-structure documentation

7.2 Slides and Course Notes

TODO: notesslides documentation

7.3 Homework, Problems and Exams

TODO: problem documentation

TODO: hwexam documentation

Part II

Documentation

Chapter 8

sTeX-Basics

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

8.1 Macros and Environments

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

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

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

8.1.1 HTML Annotations

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

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

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

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

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

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

Annotates the HTML generated by `⟨content⟩` with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

8.1.2 Babel Languages

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

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

8.1.3 Auxiliary Methods

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

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

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

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

Chapter 9

STEX-MathHub

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

9.1 Macros and Environments

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

9.1.1 Files, Paths, URIs

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

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

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

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

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

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

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

9.1.2 MathHub Archives

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

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

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

<code>\l_stex_current_repository_prop</code>
--

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

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

<code>\stex_set_current_repository:n</code>

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

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

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

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

9.1.3 Using Content in Archives

<hr/> <hr/> <code>\mhp</code> <hr/>	<code>\mhp{<i>archive-ID</i>}{<i>filename</i>}</code>
	Expands to the full path of file <i>filename</i> in repository <i>archive-ID</i> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <code>\mhinput</code> <hr/>	<code>\inputref[<i>archive-ID</i>]{<i>filename</i>}</code> Both <code>\input</code> the file <i>filename</i> in archive <i>archive-ID</i> (relative to the <code>source-</code> subdirectory). <code>\mhinput</code> does so directly. <code>\inputref</code> does so within an <code>\begingroup... \endgroup-</code> block, and skips it in <code>html</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> <code>\cmhgraphics</code> <hr/>	<i>If</i> the <code>graphicx</code> package is loaded, these macros are defined at <code>\begin{document}</code> . <code>\mhgraphics</code> takes the same arguments as <code>\includegraphics</code> , with the additional optional key <code>mhrepos</code> . It then resolves the file path in <code>\mhgraphics[mhrepos=Foo/Bar]{foo/bar.png}</code> relative to the <code>source</code> -folder of the <code>Foo/Bar</code> -archive. <code>\cmhgraphics</code> additional wraps the image in a <code>center</code> -environment.
<hr/> <hr/> <code>\lstinputmhlisting</code> <code>\clstinputmhlisting</code> <hr/>	Like <code>\mhgraphics</code> , but only defined if the <code>listings</code> -package is loaded, and with <code>\lstinputlisting</code> instead of <code>\includegraphics</code> .

Chapter 10

STEX-References

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

10.1 Macros and Environments

\STEXreftitle

\STEXreftitle{<some title>}

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

\stex_get_document_uri:

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

\l_stex_current_docns_str

Stores its result in **\l_stex_current_docns_str**

\stex_get_document_url:

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

\l_stex_current_docurl_str

Stores its result in **\l_stex_current_docurl_str**

10.1.1 Setting Reference Targets

\stex_ref_new_doc_target:n

\stex_ref_new_doc_target:n{<id>}

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

\stex_ref_new_sym_target:n

\stex_ref_new_sym_target:n{<uri>}

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

10.1.2 Using References

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

Chapter 11

STEX-Modules

This sub package contains code related to Modules

11.1 Macros and Environments

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

`\c_stex_module_<URI>_prop`

A property list with the following fields:

name The *name* of the module,

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

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

lang the module's *language*,

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

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

meta the metatheory of the module.

`\c_stex_module_<URI>_code`

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

`\c_stex_module_<URI>_constants`

The names of all constants declared in the module

`\c_stex_module_<URI>_constants`

The full URIs of all modules imported in this module

<hr/> <hr/> <code>\l_stex_current_module_str</code> <hr/> <hr/>	<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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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> <hr/> <hr/>	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 $\langle name \rangle . \langle lang \rangle . \text{tex}$ must exist in the same folder, containing a module $\langle name \rangle$.

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

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

2. Otherwise:

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

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

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

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

\l_stex_import_name_str
\l_stex_import_archive_str
\l_stex_import_path_str
\l_stex_import_ns_str

stores the result in these four variables.

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

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

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

Chapter 13

STEX-Symbols

Code related to symbol declarations and notations

13.1 Macros and Environments

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

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

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

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

Chapter 14

STEX-Terms

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

14.1 Macros and Environments

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

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

Chapter 15

TeX-Structural Features

Code related to structural features

15.1 Macros and Environments

15.1.1 Structures

`mathstructure` TODO

Chapter 16

sTeX-Statements

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

16.1 Macros and Environments

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

Chapter 17

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

17.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

17.2 The User Interface

17.2.1 Package Options

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

17.2.2 Proofs and Proof steps

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

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

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

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

17.2.3 Justifications

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

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

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

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

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

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

17.2.4 Proof Structure

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

17.2.5 Proof End Markers

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

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

17.2.6 Configuration of the Presentation

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

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

Figure 1: Configuration Hooks for Semantic Proof Markup

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

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

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

17.3 Limitations

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

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

Chapter 18

sTeX-Metatheory

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

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

18.1 Symbols

Part III
Extensions

Chapter 19

Tikzinput

19.1 Macros and Environments

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

Chapter 20

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

The `document-structure` package is part of the $\S\TeX$ collection, a version of \TeX/\LaTeX that allows to markup \TeX/\LaTeX documents semantically without leaving the document format, essentially turning \TeX/\LaTeX into a document format for mathematical knowledge management (MKM).

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

20.1 Introduction

$\S\TeX$ is a version of \TeX/\LaTeX that allows to markup \TeX/\LaTeX documents semantically without leaving the document format, essentially turning \TeX/\LaTeX into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

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

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

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

20.2 The User Interface

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

20.2.1 Package and Class Options

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

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

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

20.2.2 Document Structure

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

```

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

```

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

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

`blindfragment`

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`
`\CurrentSectionLevel`

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

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

20.2.3 Ignoring Inputs

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

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

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

`\prematurestop` For prematurely stopping the formatting of a document, \LaTeX provides the `\prematurestop` macro. It can be used everywhere in a document and ignores all input after that – backing out of the `omgroup` environment as needed. After that – and before the implicit `\end{document}` it calls the internal `\afterprematurestop`, which can be customized to do additional cleanup or e.g. print the bibliography.

`\afterprematurestop` `\prematurestop` is useful when one has a driver file, e.g. for a course taught multiple years and wants to generate course notes up to the current point in the lecture. Instead of commenting out the remaining parts, one can just move the `\prematurestop` macro. This is especially useful, if we need the rest of the file for processing, e.g. to generate a theory graph of the whole course with the already-covered parts marked up as an overview over the progress; see `import_graph.py` from the `lmhtools` utilities [LMH].

20.2.4 Structure Sharing

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

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

20.2.5 Global Variables

Text fragments and modules can be made more re-usable by the use of global variables. For instance, the admin section of a course can be made course-independent (and therefore re-usable) by using variables (actually token registers) `courseAcronym` and `courseTitle` instead of the text itself. The variables can then be set in the \LaTeX preamble of the course notes file. `\setSGvar{⟨vname⟩}{⟨text⟩}` to set the global variable `⟨vname⟩` to `⟨text⟩` and `\useSGvar{⟨vname⟩}` to reference it.

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

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

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

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

20.2.6 Colors

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

20.3 Limitations

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

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

Chapter 21

NotesSlides – Slides and Course Notes

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

21.1 Introduction

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

21.2 The User Interface

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

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

21.2.1 Package Options


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

- | | |
|---------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 21.2.2). |

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

21.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

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

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

21.2.3 Header and Footer Lines of the Slides

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

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

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

21.2.4 Frame Images

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

`\frameimage`
`\mhframeimage`

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

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

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

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

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

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

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

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

21.2.5 Colors and Highlighting

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

21.2.6 Front Matter, Titles, etc.

21.2.7 Excursions

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

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

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

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

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

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

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

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

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

21.2.8 Miscellaneous

21.3 Limitations

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

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

Chapter 22

problem.sty: An Infrastructure for formatting Problems

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

22.1 Introduction

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

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

22.2 The User Interface

22.2.1 Package Options

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

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

22.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

22.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

22.2.4 Including Problems

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

22.2.5 Reporting Metadata

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

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

22.3 Limitations

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

1. none reported yet

```

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

```

Problem 0.2 (Functions)

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

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

Problem 0.3 (Functions)

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

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

Example 7: A Problem with a multiple choice block

Chapter 23

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

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

Contents

23.1 Introduction

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

23.2 The User Interface

23.2.1 Package and Class Options

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

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

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

23.2.2 Assignments

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

23.2.3 Typesetting Exams

`multiple` Furthermore, the `hwexam` package takes the option `multiple` that allows to combine multiple assignment sheets into a compound document (the assignment sheets are treated as section, there is a table of contents, etc.).

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

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

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

23.2.4 Including Assignments

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

`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.

```
\title{320101 General Computer Science (Fall 2010)}
\begin{testheading}[duration=one hour,min=60,reqpts=27]
  Good luck to all students!
\end{testheading}
```

Name:

320101 General Computer Science (Fall 2010)

2022-03-10

You have one hour (sharp) for the test;

Write the solutions to the sheet.

The estimated time for solving this exam is 58 minutes, leaving you 2 minutes for revising your exam.

You can reach 30 points if you solve all problems. You will only need 27 points for a perfect score, i.e. 3 points are bonus points.

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

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

[illegible]

good luck

Example 8: A generated test heading.

Part IV

Implementation

Chapter 24

ST_EX -Basics Implementation

24.1 The ST_EXDocument Class

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

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

24.2 Preliminaries

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

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

24.3 Messages and logging

```

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

```

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

```

24.4 HTML Annotations

```

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

```

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

```

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

```

Conditionals for `LATeXML`:

`\if@latexml`

```

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

```

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

`\latexml_if_p:`

`\latexml_if:TF`

```

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

```

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

`\l__stex_annotate_arg_tl`
`\c__stex_annotate_emptyarg_tl`

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

```

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

```

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

`__stex_annotate_checkempty:n`

```

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

```

(End definition for `__stex_annotate_checkempty:n`.)


```

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

```

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

```

\stex_suppress_html:n Whether to (locally) produce HTML output
  105 \cs_new_protected:Nn \stex_suppress_html:n {
  106   \exp_args:Nne \use:nn {
  107     \bool_set_false:N \_stex_html_do_output_bool
  108     #1
  109   }{
  110     \stex_if_do_html:T {
  111       \bool_set_true:N \_stex_html_do_output_bool
  112     }
  113   }
  114 }

```

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

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

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

`\stex_annotate_invisible:nnn`

```

  115 \rustex_if:TF{
  116   \cs_new_protected:Nn \stex_annotate:nnn {
  117     \__stex_annotate_checkempty:n { #3 }
  118     \rustex_annotate_HTML:nn {
  119       property="stex:#1" ~
  120       resource="#2"
  121     } {
  122       \mode_if_vertical:TF{
  123         \tl_use:N \l__stex_annotate_arg_tl\par
  124       }{
  125         \tl_use:N \l__stex_annotate_arg_tl
  126       }
  127     }
  128   }
  129   \cs_new_protected:Nn \stex_annotate_invisible:n {
  130     \__stex_annotate_checkempty:n { #1 }
  131     \rustex_annotate_HTML:nn {
  132       stex:visible="false" ~
  133       style:display="none"
  134     } {
  135       \mode_if_vertical:TF{
  136         \tl_use:N \l__stex_annotate_arg_tl\par
  137       }{
  138         \tl_use:N \l__stex_annotate_arg_tl
  139       }
  140     }
  141   }

```

```

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

```

```

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

```

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

24.5 Babel Languages

```

210 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

24.6 Auxiliary Methods

\stex_deactivate_macro:Nn

```

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

```

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

\stex_reactivate_macro:N

```

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

```

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

\ignorespacesandpars

```

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

```

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

Chapter 25

STEX -MathHub Implementation

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

25.1 Generic Path Handling

We treat paths as L^AT_EX3-sequences (of the individual path segments, i.e. separated by a /-character) unix-style; i.e. a path is absolute if the sequence starts with an empty entry.

`\stex_path_from_string:Nn`

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

```

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

```

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

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

```

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

```

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

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

```

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

```

(End definition for `\c__stex_path_dot_str` and `\c__stex_path_up_str`.)

`\stex_path_canonicalize:N` Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

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

```

```

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

```

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

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

```

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

```

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

25.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

25.3 File Hooks and Tracking

```

384 <@@=stex_files>

```

We introduce hooks for file inputs that keep track of the absolute paths of files used. This will be useful to keep track of modules, their archives, namespaces etc.

Note that the absolute paths are only accurate in `\input`-statements for paths relative to the PWD, so they shouldn't be relied upon in any other setting than for \TeX -purposes.

`\g__stex_files_stack`

keeps track of file changes

```

385 \seq_gclear_new:N\g__stex_files_stack

```

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`
`\c_stex_mainfile_str`

```

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

```

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

`\g_stex_currentfile_seq`

```

389 \seq_gclear_new:N\g_stex_currentfile_seq

```

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

`\stex_filestack_push:n`

```

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

```


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

`\stex_filestack_pop:`

```

400 \cs_new_protected:Nn \stex_filestack_pop: {
401   \seq_if_empty:NF\g__stex_files_stack{
402     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
403   }
404   \seq_if_empty:NTF\g__stex_files_stack{
405     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
406   }{
407     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
408     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
409   }
410 }
```

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

Hooks for the current file:

```

411 \AddToHook{file/before}{
412   \stex_filestack_push:n{\CurrentFilePath/\CurrentFile}
413 }
414 \AddToHook{file/after}{
415   \stex_filestack_pop:
416 }
```

25.4 MathHub Repositories

417 `<@@=stex_mathhub>`

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

```

418 \str_if_empty:NTF\mathhub{
419   \sys_if_platform_windows:TF{
420     \begingroup\escapechar=-1\catcode'\=12
421     \exp_args:Nx\stex_kpsewhich:n{-expand-var~\c_percent_str MATHHUB\c_percent_str}
422     \exp_args:NNx\str_replace_all:Nnn\l_stex_kpsewhich_return_str{\c_backslash_str}/
423     \exp_args:Nnx\use:nn{\endgroup}{\str_set:Nn\exp_not:N\l_stex_kpsewhich_return_str{\l_stex_kpsewhich_return_str}}
424   }{
425     \stex_kpsewhich:n{-var-value-MATHHUB}
426   }
427   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
428 }
429 \str_if_empty:NTF\c_stex_mathhub_str{
430   \msg_warning:nn{stex}{warning/nomathhub}
431 }{
432   \stex_debug:nn{mathhub}{MathHub:~\str_use:N\c_stex_mathhub_str}
433   \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
434 }
435 }{
436   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
437   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
438     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
439       \c_stex_pwd_str/\mathhub
440     }
441   }
```

```

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

```

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

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

```

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

```

(End definition for `_stex_mathhub_do_manifest:n`.)

`\l__stex_mathhub_manifest_file_seq`

```

461 \seq_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

```

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

```

(End definition for __stex_mathhub_find_manifest:N.)

\c__stex_mathhub_manifest_ior File variable used for MANIFEST-files

```

497 \ior_new:N \c__stex_mathhub_manifest_ior

```

(End definition for \c__stex_mathhub_manifest_ior.)

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

```

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

```

```

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

```

(End definition for `_stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

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

`\l_stex_current_repository_prop`

Current MathHub repository

```

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

```

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

`\stex_in_repository:nn` Executes the code in the second argument in the context of the repository whose ID is provided as the first argument.

```

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

```

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

25.5 Using Content in Archives

`\mhpath`

```

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

```

```

612 }
613 }

```

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

`\inputref`
`\mhinput`

```

614 \newif \ifinputref \inputreffalse
615
616 \cs_new_protected:Nn \__stex_mathhub_mhinput:nn {
617   \stex_in_repository:nn {#1} {
618     \ifinputref
619       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620     \else
621       \inputreftrue
622       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
623     \inputreffalse
624   \fi
625 }
626 }
627 \NewDocumentCommand \mhinput { 0{} m}{
628   \stex_mhinput:nn{ #1 }{ #2 }
629 }
630
631 \cs_new_protected:Nn \__stex_mathhub_inputref:nn {
632   \stex_in_repository:nn {#1} {
633     \bool_lazy_any:nTF {
634       {\rustex_if_p:}
635       {\latexml_if_p:}
636     } {
637       \str_clear:N \l_tmpa_str
638       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
639         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
640       }
641       \stex_annotate_invisible:nnn{inputref}{
642         \l_tmpa_str / #2
643       }{}
644     }{
645       \begingroup
646         \inputreftrue
647         \tl_if_empty:nTF{ ##1 }{
648           \input{#2}
649         }{
650           \input{ \c_stex_mathhub_str / ##1 / source / #2 }
651         }
652       \endgroup
653     }
654   }
655 }
656 \NewDocumentCommand \inputref { 0{} m}{
657   \__stex_mathhub_inputref:nn{ #1 }{ #2 }
658 }

```

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

`\addmhbibresource`

```
659 \cs_new_protected:Nn \__stex_mathhub_mhbibresource:nn {  
660   \stex_in_repository:nn {#1} {  
661     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }  
662   }  
663 }  
664 \newcommand\addmhbibresource[2][]{  
665   \__stex_mathhub_mhbibresource:nn{ #1 }{ #2 }  
666 }
```

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

`\libinput`

```
667 \cs_new_protected:Npn \libinput #1 {  
668   \prop_if_exist:NF \l_stex_current_repository_prop {  
669     \msg_error:nnn{stex}{error/notinarchive}\libinput  
670   }  
671   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {  
672     \msg_error:nnn{stex}{error/notinarchive}\libinput  
673   }  
674   \seq_clear:N \l__stex_mathhub_libinput_files_seq  
675   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq  
676   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str  
677  
678   \bool_while_do:nn { ! \seq_if_empty_p:N \l_tmpb_seq }{  
679     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / meta-inf / lib / #1.tex}  
680     \IfFileExists{ \l_tmpa_str }{  
681       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str  
682     }{  
683       \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str  
684       \seq_put_right:No \l_tmpa_seq \l_tmpa_str  
685     }  
686  
687     \str_set:Nx \l_tmpa_str {\stex_path_to_string:N \l_tmpa_seq / lib / #1.tex}  
688     \IfFileExists{ \l_tmpa_str }{  
689       \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str  
690     }{  
691  
692       \seq_if_empty:NTF \l__stex_mathhub_libinput_files_seq {  
693         \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}  
694       }{  
695         \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {  
696           \input{ ##1 }  
697         }  
698       }  
699     }
```

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

`\libusepackage`

```
700 \NewDocumentCommand \libusepackage {0{} m} {  
701   \prop_if_exist:NF \l_stex_current_repository_prop {  
702     \msg_error:nnn{stex}{error/notinarchive}\libusepackage  
703   }
```

```

704 \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
705   \msg_error:nnn{stex}{error/notinarchive}\libusepackage
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 / #2}
713   \IfFileExists{ \l_tmpa_str.sty }{
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 / #2}
721   \IfFileExists{ \l_tmpa_str.sty }{
722     \seq_put_right:No \l__stex_mathhub_libinput_files_seq \l_tmpa_str
723   }{
724
725     \seq_if_empty:NNTF \l__stex_mathhub_libinput_files_seq {
726       \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libusepackage}{#2.sty}
727     }{
728       \int_compare:nNnTF {\seq_count:N \l__stex_mathhub_libinput_files_seq} = 1 {
729         \seq_map_inline:Nn \l__stex_mathhub_libinput_files_seq {
730           \usepackage[#1]{ ##1 }
731         }
732       }{
733         \msg_error:nnxx{stex}{error/twofiles}{\exp_not:N\libusepackage}{#2.sty}
734       }
735     }
736   }

```

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

`\mhgraphics`
`\cmhgraphics`

```

737
738 \AddToHook{begindocument}{
739   \ltx@ifpackageloaded{graphicx}{
740     \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
741     \newcommand\mhgraphics[2][{}]{%
742       \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
743       \includegraphics[#1]{\mhp\Gin@mhrepos{#2}}
744       \newcommand\cmhgraphics[2][{}]{\begin{center}\mhgraphics[#1]{#2}\end{center}}
745     }{

```

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

`\lstinputmhlisting`
`\cmlstinputmhlisting`

```

746 \ltx@ifpackageloaded{listings}{
747   \define@key{lst}{mhrepos}{\def\lst@mhrepos{#1}}
748   \newcommand\lstinputmhlisting[2][{}]{%
749     \def\lst@mhrepos{}\setkeys{lst}{#1}%
750     \lstinputlisting[#1]{\mhp\lst@mhrepos{#2}}

```



```

751     \newcommand\clstinputmhlisting[2] [] {\begin{center}\lstinputmhlisting[#1]{#2}\end{center}}
752   }{}
753 }
754
755 \end{package}

```

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

Chapter 26

STEX -References Implementation

```
756 <*package>
757
758 %%%%%%%%%% references.dtx %%%%%%%%%%
759
760 <@@=stex_refs>
761
762 Warnings and error messages
```

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

```
762 %\iow_new:N \c__stex_refs_refs_iow
763 \AddToHook{begindocument}{
764 % \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
765 }
766 \AddToHook{enddocument}{
767 % \iow_close:N \c__stex_refs_refs_iow
768 }
```

`\STEXreftitle`

```
769 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
770
771 \NewDocumentCommand \STEXreftitle { m } {
772 \tl_gset:Nx \g__stex_refs_title_tl { #1 }
773 }
```

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

26.1 Document URIs and URLs

`\l_stex_current_docns_str`

```
774 \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:`

```
775 \cs_new_protected:Nn \stex_get_document_uri: {  
776   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
777   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
778   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str  
779   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str  
780   \seq_put_right:No \l_tmpa_seq \l_tmpb_str  
781  
782   \str_clear:N \l_tmpa_str  
783   \prop_if_exist:NT \l_stex_current_repository_prop {  
784     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {  
785       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}  
786     }  
787   }  
788  
789   \str_if_empty:NTF \l_tmpa_str {  
790     \str_set:Nx \l_stex_current_docns_str {  
791       file:/\stex_path_to_string:N \l_tmpa_seq  
792     }  
793   }{  
794     \bool_set_true:N \l_tmpa_bool  
795     \bool_while_do:Nn \l_tmpa_bool {  
796       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str  
797       \exp_args:No \str_case:nnTF { \l_tmpb_str } {  
798         {source} { \bool_set_false:N \l_tmpa_bool }  
799       }{}{  
800         \seq_if_empty:NT \l_tmpa_seq {  
801           \bool_set_false:N \l_tmpa_bool  
802         }  
803       }  
804     }  
805  
806     \seq_if_empty:NTF \l_tmpa_seq {  
807       \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str  
808     }{  
809       \str_set:Nx \l_stex_current_docns_str {  
810         \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq  
811       }  
812     }  
813   }  
814 }
```

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

`\l_stex_current_docurl_str`

```
815 \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:`

```
816 \cs_new_protected:Nn \stex_get_document_url: {  
817   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq  
818   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str  
819   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
```

```

820 \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
821 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
822
823 \str_clear:N \l_tmpa_str
824 \prop_if_exist:NT \l_stex_current_repository_prop {
825   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
826     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
827       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
828     }
829   }
830 }
831
832 \str_if_empty:NTF \l_tmpa_str {
833   \str_set:Nx \l_stex_current_docurl_str {
834     file:/\stex_path_to_string:N \l_tmpa_seq
835   }
836 }{
837   \bool_set_true:N \l_tmpa_bool
838   \bool_while_do:Nn \l_tmpa_bool {
839     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
840     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
841       {source} { \bool_set_false:N \l_tmpa_bool }
842     }{}{
843       \seq_if_empty:NT \l_tmpa_seq {
844         \bool_set_false:N \l_tmpa_bool
845       }
846     }
847   }
848
849   \seq_if_empty:NTF \l_tmpa_seq {
850     \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
851   }{
852     \str_set:Nx \l_stex_current_docurl_str {
853       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
854     }
855   }
856 }
857 }

```

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

26.2 Setting Reference Targets

```

858 \str_const:Nn \c__stex_refs_url_str{URL}
859 \str_const:Nn \c__stex_refs_ref_str{REF}
860 \str_new:N \l__stex_refs_curr_label_str
861 % @currentlabel -> number
862 % @currentlabelname -> title
863 % @currentHref -> name.number <- id of some kind
864 % \theH# -> \arabic{section}
865 % \the# -> number
866 % \hyper@makecurrent{#}
867 \int_new:N \l__stex_refs_unnamed_counter_int

```

`\stex_ref_new_doc_target:n`

```

868 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
869   \stex_get_document_uri:
870   \str_clear:N \l__stex_refs_curr_label_str
871   \str_set:Nx \l_tmpa_str { #1 }
872   \str_if_empty:NT \l_tmpa_str {
873     \int_incr:N \l__stex_refs_unnamed_counter_int
874     \str_set:Nx \l_tmpa_str {REF\int_use:N \l__stex_refs_unnamed_counter_int}
875   }
876   \str_set:Nx \l__stex_refs_curr_label_str {
877     \l_stex_current_docns_str?\l_tmpa_str
878   }
879   \seq_if_exist:cF{g__stex_refs_labels_\l_tmpa_str_seq}{
880     \seq_new:c {g__stex_refs_labels_\l_tmpa_str_seq}
881   }
882   \seq_if_in:coF{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str {
883     \seq_gput_right:co{g__stex_refs_labels_\l_tmpa_str_seq}\l__stex_refs_curr_label_str
884   }
885   \stex_if_smsmode:TF {
886     \stex_get_document_url:
887     \str_gset_eq:cN {sref_url_\l__stex_refs_curr_label_str_str}\l_stex_current_docurl_str
888     \str_gset_eq:cN {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_url_str
889   }{
890     %\iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter\unexpanded\expandafter{
891     \exp_args:Nx\label{sref_\l__stex_refs_curr_label_str}
892     \immediate\write\@auxout{\stexauxadddocref{\l_stex_current_docns_str}{\l_tmpa_str}}
893     \str_gset:cx {sref_\l__stex_refs_curr_label_str_type}\c__stex_refs_ref_str
894   }
895 }

```

(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.

```

896 \cs_new_protected:Npn \stexauxadddocref #1 #2 {
897   \str_set:Nn \l_tmpa_str {#1?#2}
898   \str_gset_eq:cN{sref_#1?#2_type}\c__stex_refs_ref_str
899   \seq_if_exist:cF{g__stex_refs_labels_#2_seq}{
900     \seq_new:c {g__stex_refs_labels_#2_seq}
901   }
902   \seq_if_in:coF{g__stex_refs_labels_#2_seq}\l_tmpa_str {
903     \seq_gput_right:co{g__stex_refs_labels_#2_seq}\l_tmpa_str
904   }
905 }

```

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

```

906 \AtEndDocument{
907   \def\stexauxadddocref#1 #2 {}{}
908 }

```

`\stex_ref_new_sym_target:n`

```

909 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
910   \stex_if_smsmode:TF {
911     \str_if_exist:cF{sref_sym_#1_type}{
912       \stex_get_document_url:
913       \str_gset_eq:cN {sref_sym_url_#1_str}\l_stex_current_docurl_str

```

```

914     \str_gset_eq:cN {sref_sym_#1_type}\c__stex_refs_url_str
915   }
916 }{
917   \str_if_empty:NF \l__stex_refs_curr_label_str {
918     \str_gset_eq:cN {sref_sym_#1_label_str}\l__stex_refs_curr_label_str
919     \immediate\write\@auxout{
920       \exp_not:N\expandafter\def\exp_not:N\csname \exp_not:N\detokenize{sref_sym_#1_label_
921         \l__stex_refs_curr_label_str
922       }
923     }
924   }
925 }
926 }

```

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

26.3 Using References

```

927 \str_new:N \l__stex_refs_indocument_str

```

\sref Optional arguments:

```

928
929 \keys_define:nn { stex / sref } {
930   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
931   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
932   pre           .tl_set:N = \l__stex_refs_pre_tl ,
933   post          .tl_set:N = \l__stex_refs_post_tl ,
934 }
935 \cs_new_protected:Nn \__stex_refs_args:n {
936   \tl_clear:N \l__stex_refs_linktext_tl
937   \tl_clear:N \l__stex_refs_fallback_tl
938   \tl_clear:N \l__stex_refs_pre_tl
939   \tl_clear:N \l__stex_refs_post_tl
940   \str_clear:N \l__stex_refs_repo_str
941   \keys_set:nn { stex / sref } { #1 }
942 }

```

The actual macro:

```

943 \NewDocumentCommand \sref { 0{} m}{
944   \__stex_refs_args:n { #1 }
945   \str_if_empty:NTF \l__stex_refs_indocument_str {
946     \str_set:Nx \l_tmpa_str { #2 }
947     \exp_args:NNno \seq_set_split:Nnn \l_tmpa_seq ? \l_tmpa_str
948     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} = 1 {
949       \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str _seq}{
950         \seq_get_left:cNF {g__stex_refs_labels_\l_tmpa_str _seq} \l_tmpa_str {
951           \str_clear:N \l_tmpa_str
952         }
953       }{
954         \str_clear:N \l_tmpa_str
955       }
956     }{
957       \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
958       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str

```

```

959 \int_set:Nn \l_tmpa_int { \exp_args:Ne \str_count:n {\l_tmpb_str?\l_tmpa_str} }
960 \seq_if_exist:cTF{g__stex_refs_labels_\l_tmpa_str_seq}{
961   \str_set_eq:NN \l_tmpc_str \l_tmpa_str
962   \str_clear:N \l_tmpa_str
963   \seq_map_inline:cn {g__stex_refs_labels_\l_tmpc_str_seq} {
964     \str_if_eq:eeT { \l_tmpb_str?\l_tmpc_str }{
965       \str_range:nnn { ##1 }{ -\l_tmpa_int}{ -1 }
966     }{
967       \seq_map_break:n {
968         \str_set:Nn \l_tmpa_str { ##1 }
969       }
970     }
971   }
972 }{
973   \str_clear:N \l_tmpa_str
974 }
975 }
976 \str_if_empty:NTF \l_tmpa_str {
977   \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_lin
978 }{
979   \str_if_eq:cNTF {sref_\l_tmpa_str_type} \c__stex_refs_ref_str {
980     \tl_if_empty:NTF \l__stex_refs_linktext_tl {
981       \cs_if_exist:cTF{autoref}{
982         \l__stex_refs_pre_tl\exp_args:Nx\autoref{sref_\l_tmpa_str}\l__stex_refs_post_tl
983       }{
984         \l__stex_refs_pre_tl\exp_args:Nx\ref{sref_\l_tmpa_str}\l__stex_refs_post_tl
985       }
986     }{
987       \ltx@ifpackageloaded{hyperref}{
988         \hyperref[sref_\l_tmpa_str]\l__stex_refs_linktext_tl
989       }{
990         \l__stex_refs_linktext_tl
991       }
992     }
993   }{
994     \ltx@ifpackageloaded{hyperref}{
995       \href{\use:c{sref_url_\l_tmpa_str_str}}{\tl_if_empty:NTF \l__stex_refs_linktext_t
996     }{
997       \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs
998     }
999   }
1000 }
1001 }{
1002   % TODO
1003 }
1004 }

```

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

`\srefsym`

```

1005 \NewDocumentCommand \srefsym { 0{} m }{
1006   \stex_get_symbol:n { #2 }
1007   \__stex_refs_sym_aux:nn{##1}{\l_stex_get_symbol_uri_str}
1008 }

```

```

1009
1010 \cs_new_protected:Nn \__stex_refs_sym_aux:nn {
1011   \str_if_exist:cTF {sref_sym_#2 _label_str }{
1012     \sref[#1]{\use:c{sref_sym_#2 _label_str}}
1013   }{
1014     \__stex_refs_args:n { #1 }
1015     \str_if_empty:NTF \l__stex_refs_indocument_str {
1016       \tl_if_exist:cTF{sref_sym_#2 _type}{
1017         % doc uri in \l_tmpb_str
1018         \str_set:Nx \l_tmpa_str {\use:c{sref_sym_#2 _type}}
1019         \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
1020           % reference
1021           \tl_if_empty:NTF \l__stex_refs_linktext_tl {
1022             \cs_if_exist:cTF{autoref}{
1023               \l__stex_refs_pre_tl\autoref{sref_sym_#2}\l__stex_refs_post_tl
1024             }{
1025               \l__stex_refs_pre_tl\ref{sref_sym_#2}\l__stex_refs_post_tl
1026             }
1027           }{
1028             \ltx@ifpackageloaded{hyperref}{
1029               \hyperref[sref_sym_#2]\l__stex_refs_linktext_tl
1030             }{
1031               \l__stex_refs_linktext_tl
1032             }
1033           }
1034         }{
1035           % URL
1036           \ltx@ifpackageloaded{hyperref}{
1037             \href{\use:c{sref_sym_url_#2 _str}}{\tl_if_empty:NTF \l__stex_refs_linktext_tl \
1038           }{
1039             \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_re
1040           }
1041         }
1042       }{
1043         \tl_if_empty:NTF \l__stex_refs_linktext_tl \l__stex_refs_fallback_tl \l__stex_refs_l
1044       }
1045     }{
1046       % TODO
1047     }
1048   }
1049 }

```

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

\srefsymuri

```

1050 \cs_new_protected:Npn \srefsymuri #1 #2 {
1051   \__stex_refs_sym_aux:nn{linktext={#2}}{#1}
1052 }

```

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

```

1053 </package>

```


Chapter 27

STEX -Modules Implementation

```
1054 <*package>
1055
1056 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
1057
1058 <@@=stex_modules>
1059
1060 Warnings and error messages
1061 \msg_new:nnn{stex}{error/unknownmodule}{
1062   No~module~#1~found
1063 }
1064 \msg_new:nnn{stex}{error/syntax}{
1065   Syntax~error:~#1
1066 }
1067 \msg_new:nnn{stex}{error/siglanguage}{
1068   Module~#1~declares~signature~#2,~but~does~not~
1069   declare~its~language
1070 }
1071 \msg_new:nnn{stex}{warning/deprecated}{
1072   #1~is~deprecated;~please~use~#2~instead!
1073 }
1074 \msg_new:nnn{stex}{error/conflictingmodules}{
1075   Conflicting~imports~for~module~#1
1076 }
1077
1078 \l_stex_current_module_str The current module:
1079 \str_new:N \l_stex_current_module_str
1080
1081 (End definition for \l_stex_current_module_str. This variable is documented on page 54.)
1082
1083 \l_stex_all_modules_seq Stores all available modules
1084 \seq_new:N \l_stex_all_modules_seq
1085
1086 (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
1078 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
1079   \str_if_empty:NTF \l_stex_current_module_str
1080   \prg_return_false: \prg_return_true:
1081 }

(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
1082 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
1083   \prop_if_exist:cTF { c_stex_module_#1_prop }
1084   \prg_return_true: \prg_return_false:
1085 }

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

```

```

\stex_add_to_current_module:n
\STEXexport
Only allowed within modules:
1086 \cs_new_protected:Nn \stex_add_to_current_module:n {
1087   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
1088 }
1089 \cs_new_protected:Npn \STEXexport {
1090   \begingroup
1091   \newlinechar=-1\relax
1092   \endlinechar=-1\relax
1093   %\catcode'\ = 9\relax
1094   \expandafter\endgroup\__stex_modules_export:n
1095 }
1096 \cs_new_protected:Nn \__stex_modules_export:n {
1097   \ignorespaces #1
1098   \stex_add_to_current_module:n { \ignorespaces #1 }
1099   \stex_smsmode_do:
1100 }
1101 \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
1102 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
1103   \str_set:Nx \l_tmpa_str { #1 }
1104   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
1105 }

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

```

```

\stex_add_import_to_current_module:n
1106 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
1107   \str_set:Nx \l_tmpa_str { #1 }
1108   \exp_args:Nno
1109   \seq_if_in:cnF{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str{
1110     \seq_gput_right:co{c_stex_module_\l_stex_current_module_str _imports}\l_tmpa_str
1111   }
1112 }

```

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

`\stex_collect_imports:n`

```

1113 \cs_new_protected:Nn \stex_collect_imports:n {
1114   \seq_clear:N \l_stex_collect_imports_seq
1115   \__stex_modules_collect_imports:n {#1}
1116 }
1117 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
1118   \seq_map_inline:cn {c_stex_module_#1_imports} {
1119     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
1120       \__stex_modules_collect_imports:n { ##1 }
1121     }
1122   }
1123   \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
1124     \seq_put_right:Nx \l_stex_collect_imports_seq { #1 }
1125   }
1126 }

```

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

`\stex_do_up_to_module:n`

```

1127 \int_new:N \l__stex_modules_group_depth_int
1128 \tl_new:N \l__stex_modules_aftergroup_tl
1129 \cs_new_protected:Nn \stex_do_up_to_module:n {
1130   \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1131     #1
1132   }{
1133     #1
1134     \expandafter \tl_gset:Nn \expandafter \l__stex_modules_aftergroup_tl \expandafter { \l__
1135       \aftergroup\__stex_modules_aftergroup_do:
1136     }
1137   }
1138   \cs_new_protected:Nn \__stex_modules_aftergroup_do: {
1139     \int_compare:nNnTF \l__stex_modules_group_depth_int = \currentgrouplevel {
1140       \l__stex_modules_aftergroup_tl
1141       \tl_clear:N \l__stex_modules_aftergroup_tl
1142     }{
1143       \l__stex_modules_aftergroup_tl
1144       \aftergroup\__stex_modules_aftergroup_do:
1145     }
1146   }
1147   \cs_new_protected:Nn \stex_reset_up_to_module: {
1148
1149     \tl_gset_eq:NN \l__stex_modules_aftergroup_tl \l__stex_modules_aftergroup_outer_tl
1150   }

```

(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).

1151

(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.

```

1152 \str_new:N \l_stex_modules_ns_str
1153 \str_new:N \l_stex_modules_subpath_str
1154 \cs_new_protected:Nn \__stex_modules_compute_namespace:nN {
1155   \str_set:Nx \l_tmpa_str { #1 }
1156   \seq_set_eq:NN \l_tmpa_seq #2
1157   % split off file extension
1158   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1159   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1160   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1161   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1162
1163   \bool_set_true:N \l_tmpa_bool
1164   \bool_while_do:Nn \l_tmpa_bool {
1165     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
1166     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
1167       {source} { \bool_set_false:N \l_tmpa_bool }
1168     }{}{
1169       \seq_if_empty:NT \l_tmpa_seq {
1170         \bool_set_false:N \l_tmpa_bool
1171       }
1172     }
1173   }
1174
1175   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
1176   \str_if_empty:NTF \l_stex_modules_subpath_str {
1177     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
1178   }{
1179     \str_set:Nx \l_stex_modules_ns_str {
1180       \l_tmpa_str/\l_stex_modules_subpath_str
1181     }
1182   }
1183 }
1184
1185 \cs_new_protected:Nn \stex_modules_current_namespace: {
1186   \str_clear:N \l_stex_modules_subpath_str
1187   \prop_if_exist:NTF \l_stex_current_repository_prop {
1188     \prop_get:NnN \l_stex_current_repository_prop { ns } \l_tmpa_str
1189     \__stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1190   }{
1191     % split off file extension
1192     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1193     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1194     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1195     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1196     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1197     \str_set:Nx \l_stex_modules_ns_str {
1198       file:/\stex_path_to_string:N \l_tmpa_seq
1199     }
1200   }
1201 }

```

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

27.1 The smodule environment

smodule arguments:

```

1202 \keys_define:nn { stex / module } {
1203   title      .tl_set:N      = \smodulename ,
1204   type       .str_set_x:N    = \smodulename ,
1205   id         .str_set_x:N    = \smoduleid ,
1206   deprecate  .str_set_x:N    = \l_stex_module_deprecate_str ,
1207   ns         .str_set_x:N    = \l_stex_module_ns_str ,
1208   lang       .str_set_x:N    = \l_stex_module_lang_str ,
1209   sig        .str_set_x:N    = \l_stex_module_sig_str ,
1210   creators   .str_set_x:N    = \l_stex_module_creators_str ,
1211   contributors .str_set_x:N  = \l_stex_module_contributors_str ,
1212   meta       .str_set_x:N    = \l_stex_module_meta_str ,
1213   srccite    .str_set_x:N    = \l_stex_module_srccite_str
1214 }
1215
1216 \cs_new_protected:Nn \__stex_modules_args:n {
1217   \str_clear:N \smodulename
1218   \str_clear:N \smodulename
1219   \str_clear:N \smoduleid
1220   \str_clear:N \l_stex_module_ns_str
1221   \str_clear:N \l_stex_module_deprecate_str
1222   \str_clear:N \l_stex_module_lang_str
1223   \str_clear:N \l_stex_module_sig_str
1224   \str_clear:N \l_stex_module_creators_str
1225   \str_clear:N \l_stex_module_contributors_str
1226   \str_clear:N \l_stex_module_meta_str
1227   \str_clear:N \l_stex_module_srccite_str
1228   \keys_set:nn { stex / module } { #1 }
1229 }
1230
1231 % module parameters here? In the body?
1232

```

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

```

1233 \cs_new_protected:Nn \stex_module_setup:nn {
1234   \tl_gset_eq:NN \l__stex_modules_aftergroup_outer_tl \l__stex_modules_aftergroup_tl
1235   \tl_clear:N \l__stex_modules_aftergroup_tl
1236   \int_set:Nn \l__stex_modules_group_depth_int {\currentgrouplevel}
1237   \str_set:Nx \l_stex_module_name_str { #2 }
1238   \__stex_modules_args:n { #1 }
1239
1240   First, we set up the name and namespace of the module.
1241   Are we in a nested module?
1242
1243   \stex_if_in_module:TF {
1244     % Nested module
1245     \prop_get:cnN {c_stex_module\l_stex_current_module_str_prop}
1246     { ns } \l_stex_module_ns_str
1247     \str_set:Nx \l_stex_module_name_str {
1248       \prop_item:cn {c_stex_module\l_stex_current_module_str_prop}
1249       { name } / \l_stex_module_name_str
1250     }
1251   }
1252 }

```

```

1248 % not nested:
1249 \str_if_empty:NT \l_stex_module_ns_str {
1250   \stex_modules_current_namespace:
1251   \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1252   \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1253     / {\l_stex_module_ns_str}
1254   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1255   \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1256     \str_set:Nx \l_stex_module_ns_str {
1257       \stex_path_to_string:N \l_tmpa_seq
1258     }
1259   }
1260 }
1261 }

```

Next, we determine the language of the module:

```

1262 \str_if_empty:NT \l_stex_module_lang_str {
1263   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1264   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1265   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1266   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1267   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1268     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1269       inferred~from~file~name}
1270     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1271   }
1272 }
1273
1274 \stex_if_smsmode:F { \str_if_empty:NF \l_stex_module_lang_str {
1275   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1276   \l_tmpa_str {
1277     \ltx@ifpackageloaded{babel}{
1278       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1279     }{}
1280   } {
1281     \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1282   }
1283 }}

```

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

```

1284 \str_if_empty:NTF \l_stex_module_sig_str {
1285   \exp_args:Nnx \prop_gset_from_keyval:cn {
1286     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1287   } {
1288     name      = \l_stex_module_name_str ,
1289     ns        = \l_stex_module_ns_str ,
1290     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1291     lang      = \l_stex_module_lang_str ,
1292     sig       = \l_stex_module_sig_str ,
1293     deprecate = \l_stex_module_deprecate_str ,
1294     meta      = \l_stex_module_meta_str
1295   }
1296   \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}

```

```

1297 \seq_clear:c {c_stex_module\_l_stex_module_ns_str?\l_stex_module_name_str _constants}
1298 \tl_clear:c {c_stex_module\_l_stex_module_ns_str?\l_stex_module_name_str _code}
1299 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1300 \str_if_empty:NT \l_stex_module_meta_str {
1301   \str_set:Nx \l_stex_module_meta_str {
1302     \c_stex_metatheory_ns_str ? Metatheory
1303   }
1304 }
1305 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1306   \bool_set_true:N \l_stex_in_meta_bool
1307   \exp_args:Nx \stex_add_to_current_module:n {
1308     \bool_set_true:N \l_stex_in_meta_bool
1309     \stex_activate_module:n {\l_stex_module_meta_str}
1310     \bool_set_false:N \l_stex_in_meta_bool
1311   }
1312   \stex_activate_module:n {\l_stex_module_meta_str}
1313   \bool_set_false:N \l_stex_in_meta_bool
1314 }
1315 }{
1316   \str_if_empty:NT \l_stex_module_lang_str {
1317     \msg_error:nnxx{stex}{error/siglanguage}{
1318       \l_stex_module_ns_str?\l_stex_module_name_str
1319     }{\l_stex_module_sig_str}
1320   }
1321
1322   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1323   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1324   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1325   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1326   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1327   \str_set:Nx \l_tmpa_str {
1328     \stex_path_to_string:N \l_tmpa_seq /
1329     \l_tmpa_str . \l_stex_module_sig_str .tex
1330   }
1331   \IfFileExists \l_tmpa_str {
1332     \exp_args:No \stex_file_in_smsmode:nn { \l_tmpa_str } {
1333       \str_clear:N \l_stex_current_module_str
1334       \seq_clear:N \l_stex_all_modules_seq
1335       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1336     }
1337   }{
1338     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1339   }
1340   \stex_if_smsmode:F {
1341     \stex_activate_module:n {
1342       \l_stex_module_ns_str ? \l_stex_module_name_str
1343     }
1344   }
1345   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1346 }
1347 \str_if_empty:NF \l_stex_module_deprecate_str {
1348   \msg_warning:nnxx{stex}{warning/deprecated}{

```

```

1349     Module~\l_stex_current_module_str
1350   }{
1351     \l_stex_module_deprecate_str
1352   }
1353 }
1354 \seq_put_right:Nx \l_stex_all_modules_seq {
1355   \l_stex_module_ns_str ? \l_stex_module_name_str
1356 }
1357 }

```

(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}

1358 \cs_new_protected:Nn \__stex_modules_begin_module: {
1359   \stex_reactivate_macro:N \STEXexport
1360   \stex_reactivate_macro:N \importmodule
1361   \stex_reactivate_macro:N \symdecl
1362   \stex_reactivate_macro:N \notation
1363   \stex_reactivate_macro:N \symdef
1364
1365   \stex_debug:nn{modules}{
1366     New~module:\\
1367     Namespace:~\l_stex_module_ns_str\\
1368     Name:~\l_stex_module_name_str\\
1369     Language:~\l_stex_module_lang_str\\
1370     Signature:~\l_stex_module_sig_str\\
1371     Metatheory:~\l_stex_module_meta_str\\
1372     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1373   }
1374
1375   \stex_if_smsmode:F{
1376     \begin{stex_annotate_env} {theory} {
1377       \l_stex_module_ns_str ? \l_stex_module_name_str
1378     }
1379
1380     \stex_annotate_invisible:nnn{header}{} {
1381       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1382       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1383       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1384         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1385       }
1386       \str_if_empty:NF \smoduletype {
1387         \stex_annotate:nnn{type}{\smoduletype}{}
1388       }
1389     }
1390   }
1391   % TODO: Inherit metatheory for nested modules?
1392 }
1393 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for `__stex_modules_begin_module:.`)


```

\__stex_modules_end_module: implements \end{module}

1394 \cs_new_protected:Nn \__stex_modules_end_module: {
1395   \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module\_l_stex_current_module}}
1396 }

```

(End definition for __stex_modules_end_module:.)

The core environment

```

1397 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1398 \NewDocumentEnvironment { smodule } { 0 } { m } {
1399   \stex_module_setup:nn{#1}{#2}
1400   \par
1401   \stex_if_smsmode:F{
1402     \tl_clear:N \l_tmpa_tl
1403     \clist_map_inline:Nn \smoduletype {
1404       \tl_if_exist:cT {__stex_modules_smodule_##1_start:}{
1405         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_start:}}
1406       }
1407     }
1408     \tl_if_empty:NTF \l_tmpa_tl {
1409       \__stex_modules_smodule_start:
1410     }{
1411       \l_tmpa_tl
1412     }
1413   }
1414   \__stex_modules_begin_module:
1415   \str_if_empty:NF \smoduleid {
1416     \stex_ref_new_doc_target:n \smoduleid
1417   }
1418   \stex_smsmode_do:
1419 } {
1420   \__stex_modules_end_module:
1421   \stex_if_smsmode:F {
1422     \end{stex_annotate_env}
1423     \clist_set:No \l_tmpa_clist \smoduletype
1424     \tl_clear:N \l_tmpa_tl
1425     \clist_map_inline:Nn \l_tmpa_clist {
1426       \tl_if_exist:cT {__stex_modules_smodule_##1_end:}{
1427         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_modules_smodule_##1_end:}}
1428       }
1429     }
1430     \tl_if_empty:NTF \l_tmpa_tl {
1431       \__stex_modules_smodule_end:
1432     }{
1433       \l_tmpa_tl
1434     }
1435   }
1436 }

```

\stexpatchmodule

```

1437 \cs_new_protected:Nn \__stex_modules_smodule_start: {}
1438 \cs_new_protected:Nn \__stex_modules_smodule_end: {}
1439
1440 \newcommand\stexpatchmodule[3] [] {

```

```

1441 \str_set:Nx \l_tmpa_str{ #1 }
1442 \str_if_empty:NTF \l_tmpa_str {
1443   \tl_set:Nn \__stex_modules_smodule_start: { #2 }
1444   \tl_set:Nn \__stex_modules_smodule_end: { #3 }
1445 }{
1446   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_start:\endcsname{ #2 }
1447   \exp_after:wN \tl_set:Nn \csname __stex_modules_smodule_#1_end:\endcsname{ #3 }
1448 }
1449 }

```

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

27.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1450 \NewDocumentCommand \STEXModule { m } {
1451   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1452   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1453   \tl_set:Nn \l_tmpa_tl {
1454     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1455   }
1456   \seq_map_inline:Nn \l_stex_all_modules_seq {
1457     \str_set:Nn \l_tmpb_str { ##1 }
1458     \str_if_eq:eeT { \l_tmpa_str } {
1459       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1460     } {
1461       \seq_map_break:n {
1462         \tl_set:Nn \l_tmpa_tl {
1463           \stex_invoke_module:n { ##1 }
1464         }
1465       }
1466     }
1467   }
1468   \l_tmpa_tl
1469 }
1470
1471 \cs_new_protected:Nn \stex_invoke_module:n {
1472   \stex_debug:nn{modules}{Invoking~module~#1}
1473   \peek_charcode_remove:NTF ! {
1474     \__stex_modules_invoke_uri:nN { #1 }
1475   } {
1476     \peek_charcode_remove:NTF ? {
1477       \__stex_modules_invoke_symbol:nn { #1 }
1478     } {
1479       \msg_error:nnx{stex}{error/syntax}{
1480         ?~or~!~expected~after~
1481         \c_backslash_str STEXModule{#1}
1482       }
1483     }
1484   }
1485 }
1486
1487 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {

```

```

1488 \str_set:Nn #2 { #1 }
1489 }
1490
1491 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1492 \stex_invoke_symbol:n{#1?#2}
1493 }

```

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

`\stex_activate_module:n`

```

1494 \bool_new:N \l_stex_in_meta_bool
1495 \bool_set_false:N \l_stex_in_meta_bool
1496 \cs_new_protected:Nn \stex_activate_module:n {
1497 \stex_debug:nn{modules}{Activating~module~#1}
1498 \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1499 \msg_error:nnn{stex}{error/conflictingmodules}{ #1 }
1500 }
1501 \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1502 \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1503 \use:c{ c_stex_module_#1_code }
1504 }
1505 }

```

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

```

1506 </package>

```

Chapter 28

STEX -Module Inheritance Implementation

```
1507 <*package>
1508
1509 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1510
```

28.1 SMS Mode

```
1511 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1512 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1513 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1514 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1515
1516 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1517   \makeatletter
1518   \makeatother
1519   \ExplSyntaxOn
1520   \ExplSyntaxOff
1521   \rustexBREAK
1522 }
1523
1524 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1525   \symdef
1526   \importmodule
1527   \notation
1528   \symdecl
1529   \STEXexport
1530   \inlineass
1531   \inlinedef
1532   \inlineex
1533   \endinput
1534   \setnotation
```

```

1535 \copynotation
1536 }
1537
1538 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1539   \tl_to_str:n {
1540     smodule,
1541     copymodule,
1542     interpretmodule,
1543     sdefinition,
1544     sexample,
1545     sassertion,
1546     sparagraph
1547   }
1548 }

```

(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`

```

1549 \bool_new:N \g__stex_smsmode_bool
1550 \bool_set_false:N \g__stex_smsmode_bool
1551 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1552   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1553 }

```

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

`_stex_smsmode_in_smsmode:nn`

```

1554 \cs_new_protected:Nn \_stex_smsmode_in_smsmode:nn {
1555   \vbox_set:Nn \l_tmpa_box {
1556     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1557     \bool_gset_true:N \g__stex_smsmode_bool
1558     #2
1559     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1560   }
1561   \box_clear:N \l_tmpa_box
1562 }

```

(End definition for `_stex_smsmode_in_smsmode:nn`.)

`\stex_file_in_smsmode:nn`

```

1563 \quark_new:N \q__stex_smsmode_break
1564
1565 \NewDocumentCommand \_stex_smsmode_importmodule: { 0{} m } {
1566   \seq_gput_right:Nn \l__stex_smsmode_importmodules_seq {{#1}{#2}}
1567   \stex_smsmode_do:
1568 }
1569
1570 \cs_new_protected:Nn \stex_file_in_smsmode:nn {
1571   \stex_filestack_push:n{#1}
1572   \seq_gclear:N \l__stex_smsmode_importmodules_seq
1573   % ----- new -----
1574   \_stex_smsmode_in_smsmode:nn{#1}{
1575     \let\importmodule\_stex_smsmode_importmodule:
1576     \seq_clear:N \g_stex_smsmode_allowedenvs_seq

```

```

1577 \tl_clear:N \g_stex_smsmode_allowedmacros_tl
1578 \tl_clear:N \g_stex_smsmode_allowedmacros_escape_tl
1579 \tl_put_right:Nn \g_stex_smsmode_allowedmacros_escape_tl {\importmodule}
1580 \everyeof{\q__stex_smsmode_break\noexpand}
1581 \expandafter\expandafter\expandafter
1582 \stex_smsmode_do:
1583 \csname @ @ input\endcsname "#1"\relax
1584 }
1585 % ----- new -----
1586 \__stex_smsmode_in_smsmode:nn{#1} {
1587 #2
1588 % ----- new -----
1589 \begingroup
1590 \stex_debug:nn{smsmode}{Here:~\seq_use:Nn\l__stex_smsmode_importmodules_seq, }
1591 \seq_map_inline:Nn \l__stex_smsmode_importmodules_seq {
1592 \stex_import_module_uri:nn ##1
1593 \stex_import_require_module:nnnn
1594 \l_stex_import_ns_str
1595 \l_stex_import_archive_str
1596 \l_stex_import_path_str
1597 \l_stex_import_name_str
1598 }
1599 \endgroup
1600 \stex_debug:nn{smsmode}{Actually~loading~file~#1}
1601 % ----- new -----
1602 \everyeof{\q__stex_smsmode_break\noexpand}
1603 \expandafter\expandafter\expandafter
1604 \stex_smsmode_do:
1605 \csname @ @ input\endcsname "#1"\relax
1606 }
1607 \stex_filestack_pop:
1608 }

```

(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:

```

1609 \cs_new_protected:Npn \stex_smsmode_do: {
1610 \stex_if_smsmode:T {
1611 \__stex_smsmode_do:w
1612 }
1613 }
1614 \cs_new_protected:Npn \__stex_smsmode_do:w #1 {
1615 \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #1 } }{
1616 \expandafter\if\expandafter\relax\noexpand#1
1617 \expandafter\__stex_smsmode_do_aux:N\expandafter#1
1618 \else\expandafter\__stex_smsmode_do:w\fi
1619 }{
1620 \__stex_smsmode_do:w % #1
1621 }
1622 }
1623 \cs_new_protected:Nn \__stex_smsmode_do_aux:N {
1624 \cs_if_eq:NNF #1 \q__stex_smsmode_break {
1625 \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_tl {#1} {

```

```

1626     #1\__stex_smsmode_do:w
1627   }{
1628     \tl_if_in:NnTF \g_stex_smsmode_allowedmacros_escape_tl {#1} {
1629       #1
1630     }{
1631       \cs_if_eq:NNTF \begin #1 {
1632         \__stex_smsmode_check_begin:n
1633       }{
1634         \cs_if_eq:NNTF \end #1 {
1635           \__stex_smsmode_check_end:n
1636         }{
1637           \__stex_smsmode_do:w
1638         }
1639       }
1640     }
1641   }
1642 }
1643 }
1644
1645 \cs_new_protected:Nn \__stex_smsmode_check_begin:n {
1646   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1647     \begin{#1}
1648   }{
1649     \__stex_smsmode_do:w
1650   }
1651 }
1652 \cs_new_protected:Nn \__stex_smsmode_check_end:n {
1653   \seq_if_in:NxTF \g_stex_smsmode_allowedenvs_seq { \detokenize{#1} }{
1654     \end{#1}\__stex_smsmode_do:w
1655   }{
1656     \str_if_eq:nnTF{#1}{document}{\endinput}{\__stex_smsmode_do:w}
1657   }
1658 }

```

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

28.2 Inheritance

```

1659 <@@=stex_importmodule>

\stex_import_module_uri:nn

1660 \cs_new_protected:Nn \stex_import_module_uri:nn {
1661   \str_set:Nx \l_stex_import_archive_str { #1 }
1662   \str_set:Nn \l_stex_import_path_str { #2 }
1663
1664   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l_stex_import_path_str }
1665   \seq_pop_right:NN \l_tmpb_seq \l_stex_import_name_str
1666   \str_set:Nx \l_stex_import_path_str { \seq_use:Nn \l_tmpb_seq ? }
1667
1668   \stex_modules_current_namespace:
1669   \bool_lazy_all:nTF {
1670     {\str_if_empty_p:N \l_stex_import_archive_str}
1671     {\str_if_empty_p:N \l_stex_import_path_str}
1672     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l_stex_import_name_str } }

```

```

1673 }{
1674   \str_set_eq:NN \l_stex_import_path_str \l_stex_modules_subpath_str
1675   \str_set_eq:NN \l_stex_import_ns_str \l_stex_module_ns_str
1676 }{
1677   \str_if_empty:NT \l_stex_import_archive_str {
1678     \prop_if_exist:NT \l_stex_current_repository_prop {
1679       \prop_get:NnN \l_stex_current_repository_prop { id } \l_stex_import_archive_str
1680     }
1681   }
1682   \str_if_empty:NTF \l_stex_import_archive_str {
1683     \str_if_empty:NF \l_stex_import_path_str {
1684       \str_set:Nx \l_stex_import_ns_str {
1685         \l_stex_module_ns_str / \l_stex_import_path_str
1686       }
1687     }
1688   }{
1689     \stex_require_repository:n \l_stex_import_archive_str
1690     \prop_get:cnN { c_stex_mathhub_\l_stex_import_archive_str _manifest_prop } { ns }
1691     \l_stex_import_ns_str
1692     \str_if_empty:NF \l_stex_import_path_str {
1693       \str_set:Nx \l_stex_import_ns_str {
1694         \l_stex_import_ns_str / \l_stex_import_path_str
1695       }
1696     }
1697   }
1698 }
1699 }

```

(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
1700 \str_new:N \l_stex_import_name_str
1701 \str_new:N \l_stex_import_archive_str
1702 \str_new:N \l_stex_import_path_str
1703 \str_new:N \l_stex_import_ns_str

```

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

```

\stex_import_require_module:nmmn {{(ns)} {(archive-ID)} {(path)} {(name)}}
1704 \cs_new_protected:Nn \stex_import_require_module:nmmn {
1705   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1706
1707     % archive
1708     \str_set:Nx \l_tmpa_str { #2 }
1709     \str_if_empty:NTF \l_tmpa_str {
1710       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1711     } {
1712       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1713       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1714       \seq_put_right:Nn \l_tmpa_seq { source }
1715     }
1716
1717     % path
1718     \str_set:Nx \l_tmpb_str { #3 }

```



```

1719 \str_if_empty:NTF \l_tmpb_str {
1720   \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1721
1722   \ltx@ifpackageloaded{babel} {
1723     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1724       { \language } \l_tmpb_str {
1725       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1726     }
1727   } {
1728     \str_clear:N \l_tmpb_str
1729   }
1730
1731   \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1732   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1733     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1734   }{
1735     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1736     \IfFileExists{ \l_tmpa_str.tex }{
1737       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1738     }{
1739       % try english as default
1740       \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1741       \IfFileExists{ \l_tmpa_str.en.tex }{
1742         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1743       }{
1744         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1745       }
1746     }
1747   }
1748
1749 } {
1750   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1751   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1752
1753   \ltx@ifpackageloaded{babel} {
1754     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1755       { \language } \l_tmpb_str {
1756       \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1757     }
1758   } {
1759     \str_clear:N \l_tmpb_str
1760   }
1761
1762   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1763
1764   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1765   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1766     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1767   }{
1768     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1769     \IfFileExists{ \l_tmpa_str/#4.tex }{
1770       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1771     }{
1772       % try english as default

```

```

1773 \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1774 \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1775   \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1776 }{
1777   \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1778   \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1779     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1780   }{
1781     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1782     \IfFileExists{ \l_tmpa_str.tex }{
1783       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1784     }{
1785       % try english as default
1786       \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1787       \IfFileExists{ \l_tmpa_str.en.tex }{
1788         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1789       }{
1790         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1791       }
1792     }
1793   }
1794 }
1795 }
1796 }
1797 }
1798
1799 \exp_args:No \stex_file_in_smsmode:nn { \g__stex_importmodule_file_str } {
1800   \seq_clear:N \l_stex_all_modules_seq
1801   \str_clear:N \l_stex_current_module_str
1802   \str_set:Nx \l_tmpb_str { #2 }
1803   \str_if_empty:NF \l_tmpb_str {
1804     \stex_set_current_repository:n { #2 }
1805   }
1806   \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1807 }
1808
1809 \stex_if_module_exists:nF { #1 ? #4 } {
1810   \msg_error:nnx{stex}{error/unknownmodule}{
1811     #1?#4~(in~file~\g__stex_importmodule_file_str)
1812   }
1813 }
1814 }
1815 \stex_activate_module:n { #1 ? #4 }
1816 }

```

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

`\importmodule`

```

1817 \NewDocumentCommand \importmodule { 0{} m } {
1818   \stex_import_module_uri:nn { #1 } { #2 }
1819   \stex_debug:nn{modules}{Importing~module:~
1820     \l_stex_import_ns_str ? \l_stex_import_name_str
1821   }
1822   \stex_import_require_module:nnnn

```

```

1823 { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1824 { \l_stex_import_path_str } { \l_stex_import_name_str }
1825 \stex_if_smsmode:F {
1826   \stex_annotate_invisible:nnn
1827   {import} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1828 }
1829 \exp_args:Nx \stex_add_to_current_module:n {
1830   \stex_import_require_module:nnnn
1831   { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1832   { \l_stex_import_path_str } { \l_stex_import_name_str }
1833 }
1834 \exp_args:Nx \stex_add_import_to_current_module:n {
1835   \l_stex_import_ns_str ? \l_stex_import_name_str
1836 }
1837 \stex_smsmode_do:
1838 \ignorespacesandpars
1839 }
1840 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1841 \NewDocumentCommand \usemodule { 0{} m } {
1842   \stex_if_smsmode:F {
1843     \stex_import_module_uri:nn { #1 } { #2 }
1844     \stex_import_require_module:nnnn
1845     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
1846     { \l_stex_import_path_str } { \l_stex_import_name_str }
1847     \stex_annotate_invisible:nnn
1848     {usemodule} {\l_stex_import_ns_str ? \l_stex_import_name_str} {}
1849   }
1850   \stex_smsmode_do:
1851   \ignorespacesandpars
1852 }

```

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

```

1853 \endpackage

```

Chapter 29

STEX -Symbols Implementation

```
1854 <*package>
1855
1856 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1857
      Warnings and error messages
1858 \msg_new:nnn{stex}{error/wrongargs}{
1859   args~value~in~symbol~declaration~for~#1~
1860   needs~to~be~i,~a,~b~or~B,~but~#2~given
1861 }
1862 \msg_new:nnn{stex}{error/unknownsymbol}{
1863   No~symbol~#1~found!
1864 }
1865 \msg_new:nnn{stex}{error/seqlength}{
1866   Expected~#1~arguments;~got~#2!
1867 }
```

29.1 Symbol Declarations

```
1868 <@@=stex_symdecl>
\stex_all_symbols:n Map over all available symbols
1869 \cs_new_protected:Nn \stex_all_symbols:n {
1870   \def \__stex_symdecl_all_symbols_cs ##1 {#1}
1871   \seq_map_inline:Nn \l_stex_all_modules_seq {
1872     \seq_map_inline:cn{c_stex_module_##1_constants}{
1873       \__stex_symdecl_all_symbols_cs{##1?####1}
1874     }
1875   }
1876 }

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

\STEXsymbol
1877 \NewDocumentCommand \STEXsymbol { m } {
1878   \stex_get_symbol:n { #1 }
```

```

1879 \exp_args:No
1880 \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1881 }

```

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

`symdecl` arguments:

```

1882 \keys_define:nn { stex / symdecl } {
1883   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1884   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1885   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1886   type      .tl_set:N = \l_stex_symdecl_type_tl ,
1887   deprecate .str_set_x:N = \l_stex_symdecl_deprecate_str ,
1888   align     .str_set:N = \l_stex_symdecl_align_str , % TODO(?)
1889   gfc       .str_set:N = \l_stex_symdecl_gfc_str , % TODO(?)
1890   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1891   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
1892   assoc     .choices:nn =
1893     {bin,binl,binr,pre,conj,pwconj}
1894     {\str_set:Nx \l_stex_symdecl_astype_str {\l_keys_choice_tl}}
1895 }
1896
1897 \bool_new:N \l_stex_symdecl_make_macro_bool
1898
1899 \cs_new_protected:Nn \__stex_symdecl_args:n {
1900   \str_clear:N \l_stex_symdecl_name_str
1901   \str_clear:N \l_stex_symdecl_args_str
1902   \str_clear:N \l_stex_symdecl_deprecate_str
1903   \str_clear:N \l_stex_symdecl_astype_str
1904   \bool_set_false:N \l_stex_symdecl_local_bool
1905   \tl_clear:N \l_stex_symdecl_type_tl
1906   \tl_clear:N \l_stex_symdecl_definiens_tl
1907
1908   \keys_set:nn { stex / symdecl } { #1 }
1909 }

```

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

```

1910
1911 \NewDocumentCommand \symdecl { s m O{} } {
1912   \__stex_symdecl_args:n { #3 }
1913   \IfBooleanTF #1 {
1914     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1915   } {
1916     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1917   }
1918   \stex_symdecl_do:n { #2 }
1919   \stex_smsmode_do:
1920 }
1921
1922 \cs_new_protected:Nn \stex_symdecl_do:nn {
1923   \__stex_symdecl_args:n{#1}
1924   \bool_set_false:N \l_stex_symdecl_make_macro_bool
1925   \stex_symdecl_do:n{#2}
1926 }

```

```

1927
1928 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1929 \cs_new_protected:Nn \stex_symdecl_do:n {
1930   \stex_if_in_module:F {
1931     % TODO throw error? some default namespace?
1932   }
1933
1934   \str_if_empty:NT \l_stex_symdecl_name_str {
1935     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1936   }
1937
1938   \prop_if_exist:cT { l_stex_symdecl_
1939     \l_stex_current_module_str ?
1940     \l_stex_symdecl_name_str
1941   }_prop
1942   {
1943     % TODO throw error (beware of circular dependencies)
1944   }
1945
1946   \prop_clear:N \l_tmpa_prop
1947   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1948   \seq_clear:N \l_tmpa_seq
1949   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1950   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1951
1952   \str_if_empty:NT \l_stex_symdecl_deprecate_str {
1953     \str_if_empty:NF \l_stex_module_deprecate_str {
1954       \str_set_eq:NN \l_stex_symdecl_deprecate_str \l_stex_module_deprecate_str
1955     }
1956   }
1957   \prop_put:Nno \l_tmpa_prop { deprecate } \l_stex_symdecl_deprecate_str
1958
1959   \exp_args:No \stex_add_constant_to_current_module:n {
1960     \l_stex_symdecl_name_str
1961   }
1962
1963   % arity/args
1964   \int_zero:N \l_tmpb_int
1965
1966   \bool_set_true:N \l_tmpa_bool
1967   \str_map_inline:Nn \l_stex_symdecl_args_str {
1968     \token_case_meaning:NnF ##1 {
1969       0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1970       {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1971       {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1972       {\tl_to_str:n a} {
1973         \bool_set_false:N \l_tmpa_bool
1974         \int_incr:N \l_tmpb_int
1975       }
1976       {\tl_to_str:n B} {

```

```

1977     \bool_set_false:N \l_tmpa_bool
1978     \int_incr:N \l_tmpb_int
1979   }
1980 }{
1981   \msg_error:nnxx{stex}{error/wrongargs}{
1982     \l_stex_current_module_str ?
1983     \l_stex_symdecl_name_str
1984   }{##1}
1985 }
1986 }
1987 \bool_if:NTF \l_tmpa_bool {
1988   % possibly numeric
1989   \str_if_empty:NTF \l_stex_symdecl_args_str {
1990     \prop_put:Nnn \l_tmpa_prop { args } {}
1991     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1992   }{
1993     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1994     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1995     \str_clear:N \l_tmpa_str
1996     \int_step_inline:nn \l_tmpa_int {
1997       \str_put_right:Nn \l_tmpa_str i
1998     }
1999     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
2000   }
2001 } {
2002   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
2003   \prop_put:Nnx \l_tmpa_prop { arity }
2004     { \str_count:N \l_stex_symdecl_args_str }
2005 }
2006 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2007
2008 \tl_if_empty:NTF \l_stex_symdecl_definiens_tl {
2009   \prop_put:Nnx \l_tmpa_prop { defined }{ false }
2010 }{
2011   \prop_put:Nnx \l_tmpa_prop { defined }{ true }
2012 }
2013
2014 % semantic macro
2015
2016 \bool_if:NT \l_stex_symdecl_make_macro_bool {
2017   \exp_args:Nx \stex_do_up_to_module:n {
2018     \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2019       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2020     }}
2021   }
2022
2023   \bool_if:NF \l_stex_symdecl_local_bool {
2024     \exp_args:Nx \stex_add_to_current_module:n {
2025       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
2026         \l_stex_current_module_str ? \l_stex_symdecl_name_str
2027       } }
2028     }
2029   }
2030 }

```

```

2031
2032 \stex_debug:nn{symbols}{New~symbol:~
2033   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
2034   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
2035   Args:~\prop_item:Nn \l_tmpa_prop { args }^^J
2036   Definiens:~\exp_not:o { \l_stex_symdecl_definiens_tl }
2037 }
2038
2039 % circular dependencies require this:
2040
2041 \prop_if_exist:cF {
2042   \l_stex_symdecl_
2043   \l_stex_current_module_str ? \l_stex_symdecl_name_str
2044   _prop
2045 } {
2046   \exp_args:Nx \stex_do_up_to_module:n {
2047     \prop_set_from_keyval:cn {
2048       \l_stex_symdecl_
2049       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2050       _prop
2051     } {\prop_to_keyval:N \l_tmpa_prop}
2052     \seq_clear:c {
2053       \l_stex_symdecl_
2054       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2055       _notations
2056     }
2057   }
2058 }
2059
2060
2061
2062 \bool_if:NF \l_stex_symdecl_local_bool {
2063   \exp_args:Nx
2064   \stex_add_to_current_module:n {
2065     \seq_clear:c {
2066       \l_stex_symdecl_
2067       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2068       _notations
2069     }
2070     \prop_set_from_keyval:cn {
2071       \l_stex_symdecl_
2072       \l_stex_current_module_str ? \l_stex_symdecl_name_str
2073       _prop
2074     } {
2075       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
2076       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
2077       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
2078       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
2079       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
2080       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
2081     }
2082   }
2083 }
2084

```



```

2085 \stex_if_smsmode:F {
2086 % \exp_args:Nx \stex_do_up_to_module:n {
2087 % \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
2088 % \l_stex_current_module_str ? \l_stex_symdecl_name_str
2089 % }
2090 % }
2091 \stex_if_do_html:T {
2092 \stex_annotate_invisible:nnn {symdecl} {
2093 \l_stex_current_module_str ? \l_stex_symdecl_name_str
2094 } {
2095 \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{${\l_st
2096 \stex_annotate_invisible:nnn{args}}{}{
2097 \prop_item:Nn \l_tmpa_prop { args }
2098 }
2099 \stex_annotate_invisible:nnn{macroname}{#1}{}
2100 \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
2101 \stex_annotate_invisible:nnn{definiens}{}
2102 {${\l_stex_symdecl_definiens_tl$}
2103 }
2104 \str_if_empty:NF \l_stex_symdecl_assoc_type_str {
2105 \stex_annotate_invisible:nnn{assoc_type}{\l_stex_symdecl_assoc_type_str}{}
2106 }
2107 }
2108 }
2109 }
2110 }

```

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

`\stex_get_symbol:n`

```

2111 \str_new:N \l_stex_get_symbol_uri_str
2112
2113 \cs_new_protected:Nn \stex_get_symbol:n {
2114 \tl_if_head_eq_catcode:nNTF { #1 } \relax {
2115 \tl_set:Nn \l_tmpa_tl { #1 }
2116 \__stex_symdecl_get_symbol_from_cs:
2117 }{
2118 % argument is a string
2119 % is it a command name?
2120 \cs_if_exist:cTF { #1 }{
2121 \cs_set_eq:Nc \l_tmpa_tl { #1 }
2122 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
2123 \str_if_empty:NTF \l_tmpa_str {
2124 \exp_args:Nx \cs_if_eq:NNTF {
2125 \tl_head:N \l_tmpa_tl
2126 } \stex_invoke_symbol:n {
2127 \__stex_symdecl_get_symbol_from_cs:
2128 }{
2129 \__stex_symdecl_get_symbol_from_string:n { #1 }
2130 }
2131 } {
2132 \__stex_symdecl_get_symbol_from_string:n { #1 }
2133 }
2134 }{

```

```

2135     % argument is not a command name
2136     \__stex_symdecl_get_symbol_from_string:n { #1 }
2137     % \l_stex_all_symbols_seq
2138   }
2139 }
2140 \str_if_eq:eeF {
2141   \prop_item:cn {
2142     l_stex_symdecl\l_stex_get_symbol_uri_str _prop
2143   }{ deprecate }
2144 }{}{
2145   \msg_warning:nnxx{stex}{warning/deprecated}{
2146     Symbol~\l_stex_get_symbol_uri_str
2147   }{
2148     \prop_item:cn {l_stex_symdecl\l_stex_get_symbol_uri_str _prop}{ deprecate }
2149   }
2150 }
2151 }
2152
2153 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
2154   \tl_set:Nn \l_tmpa_tl {
2155     \msg_error:nnn{stex}{error/unknownsymbol}{#1}
2156   }
2157   \str_set:Nn \l_tmpa_str { #1 }
2158   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2159
2160   \stex_all_symbols:n {
2161     \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1}{-\l_tmpa_int}{-1}}{
2162       \seq_map_break:n{ \seq_map_break:n{
2163         \tl_set:Nn \l_tmpa_tl {
2164           \str_set:Nn \l_stex_get_symbol_uri_str { ##1 }
2165         }
2166       }}
2167     }
2168   }
2169
2170   \l_tmpa_tl
2171 }
2172
2173 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs: {
2174   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2175     { \tl_tail:N \l_tmpa_tl }
2176   \tl_if_single:NTF \l_tmpa_tl {
2177     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2178       \exp_after:wN \str_set:Nn \exp_after:wN
2179         \l_stex_get_symbol_uri_str \l_tmpa_tl
2180     }{
2181       % TODO
2182       % tail is not a single group
2183     }
2184   }{
2185     % TODO
2186     % tail is not a single group
2187   }
2188 }

```

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

29.2 Notations

2189 `<@@=stex_notation>`

```

notation arguments:
2190 \keys_define:nn { stex / notation } {
2191   lang      .tl_set_x:N = \l__stex_notation_lang_str ,
2192   variant   .tl_set_x:N = \l__stex_notation_variant_str ,
2193   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2194   op        .tl_set:N   = \l__stex_notation_op_tl ,
2195   primary   .bool_set:N = \l__stex_notation_primary_bool ,
2196   primary   .default:n  = {true} ,
2197   unknown   .code:n     = \str_set:Nx
2198               \l__stex_notation_variant_str \l_keys_key_str
2199 }
2200
2201 \cs_new_protected:Nn \_stex_notation_args:n {
2202   \str_clear:N \l__stex_notation_lang_str
2203   \str_clear:N \l__stex_notation_variant_str
2204   \str_clear:N \l__stex_notation_prec_str
2205   \tl_clear:N \l__stex_notation_op_tl
2206   \bool_set_false:N \l__stex_notation_primary_bool
2207
2208   \keys_set:nn { stex / notation } { #1 }
2209 }
```

`\notation`

```

2210 \NewDocumentCommand \notation { s m O{} } {
2211   \_stex_notation_args:n { #3 }
2212   \tl_clear:N \l_stex_symdecl_definiens_tl
2213   \stex_get_symbol:n { #2 }
2214   \tl_set:Nn \l_stex_notation_after_do_tl {
2215     \__stex_notation_final:
2216     \IfBooleanTF#1{
2217       \stex_setnotation:n {\l_stex_get_symbol_uri_str}
2218     }{}
2219     \stex_smsmode_do:\ignorespacesandpars
2220   }
2221   \stex_notation_do:nnnnn
2222     { \prop_item:cn {l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { args } }
2223     { \prop_item:cn { l_stex_symdecl\_l_stex_get_symbol_uri_str_prop } { arity } }
2224     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2225     { \l__stex_notation_prec_str }
2226   }
2227   \stex_deactivate_macro:Nn \notation {module~environments}
```

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

`\stex_notation_do:nnnnn`

```

2228 \seq_new:N \l__stex_notation_precedences_seq
2229 \tl_new:N \l__stex_notation_opprec_tl
2230 \int_new:N \l__stex_notation_currarg_int
```

```

2231 \tl_new:N \stex_symbol_after_invokation_tl
2232
2233 \cs_new_protected:Nn \stex_notation_do:nnnnn {
2234   \let\l_stex_current_symbol_str\relax
2235   \seq_clear:N \l__stex_notation_precedences_seq
2236   \tl_clear:N \l__stex_notation_opprec_tl
2237   \str_set:Nx \l__stex_notation_args_str { #1 }
2238   \str_set:Nx \l__stex_notation_arity_str { #2 }
2239   \str_set:Nx \l__stex_notation_suffix_str { #3 }
2240   \str_set:Nx \l__stex_notation_prec_str { #4 }
2241
2242   % precedences
2243   \str_if_empty:NTF \l__stex_notation_prec_str {
2244     \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2245       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2246     }{
2247       \tl_set:Nn \l__stex_notation_opprec_tl { 0 }
2248     }
2249   } {
2250     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2251       \tl_set:No \l__stex_notation_opprec_tl { \neginfprec }
2252       \int_step_inline:nn { \l__stex_notation_arity_str } {
2253         \exp_args:NNo
2254         \seq_put_right:Nn \l__stex_notation_precedences_seq { \infprec }
2255       }
2256     }{
2257       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2258       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2259         \tl_set:No \l__stex_notation_opprec_tl { \l_tmpa_str }
2260         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2261           \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2262             \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2263           \seq_map_inline:Nn \l_tmpa_seq {
2264             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2265           }
2266         }
2267       }{
2268         \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2269           \tl_set:No \l__stex_notation_opprec_tl { \infprec }
2270         }{
2271           \tl_set:No \l__stex_notation_opprec_tl { 0 }
2272         }
2273       }
2274     }
2275   }
2276
2277   \seq_set_eq:NN \l_tmpa_seq \l__stex_notation_precedences_seq
2278   \int_step_inline:nn { \l__stex_notation_arity_str } {
2279     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2280       \exp_args:NNo
2281       \seq_put_right:No \l__stex_notation_precedences_seq {
2282         \l__stex_notation_opprec_tl
2283       }
2284     }
  
```

```

2285 }
2286 \tl_clear:N \l_stex_notation_dummyargs_tl
2287
2288 \int_compare:nNnTF \l__stex_notation_arity_str = 0 {
2289   \exp_args:NNe
2290   \cs_set:Npn \l_stex_notation_macrocode_cs {
2291     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }
2292     { \l__stex_notation_suffix_str }
2293     { \l__stex_notation_opprec_tl }
2294     { \exp_not:n { #5 } }
2295   }
2296   \l_stex_notation_after_do_tl
2297 }{
2298   \str_if_in:NnTF \l__stex_notation_args_str b {
2299     \exp_args:Nne \use:nn
2300     {
2301       \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2302       \cs_set:Npn \l__stex_notation_arity_str } { {
2303         \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2304         { \l__stex_notation_suffix_str }
2305         { \l__stex_notation_opprec_tl }
2306         { \exp_not:n { #5 } }
2307       }}
2308   }{
2309     \str_if_in:NnTF \l__stex_notation_args_str B {
2310       \exp_args:Nne \use:nn
2311       {
2312         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2313         \cs_set:Npn \l__stex_notation_arity_str } { {
2314           \_stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2315           { \l__stex_notation_suffix_str }
2316           { \l__stex_notation_opprec_tl }
2317           { \exp_not:n { #5 } }
2318         } }
2319     }{
2320       \exp_args:Nne \use:nn
2321       {
2322         \cs_generate_from_arg_count:NNnn \l_stex_notation_macrocode_cs
2323         \cs_set:Npn \l__stex_notation_arity_str } { {
2324           \_stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2325           { \l__stex_notation_suffix_str }
2326           { \l__stex_notation_opprec_tl }
2327           { \exp_not:n { #5 } }
2328         } }
2329     }
2330   }
2331
2332   \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2333   \int_zero:N \l__stex_notation_currarg_int
2334   \seq_set_eq:NN \l__stex_notation_remaining_precs_seq \l__stex_notation_precedences_seq
2335   \__stex_notation_arguments:
2336 }
2337 }

```

(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

```

2338 \cs_new_protected:Nn \__stex_notation_arguments: {
2339   \int_incr:N \l__stex_notation_currarg_int
2340   \str_if_empty:NTF \l__stex_notation_remaining_args_str {
2341     \l_stex_notation_after_do_tl
2342   }{
2343     \str_set:Nx \l_tmpa_str { \str_head:N \l__stex_notation_remaining_args_str }
2344     \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_remaini
2345     \str_if_eq:VnTF \l_tmpa_str a {
2346       \__stex_notation_argument_assoc:n
2347     }{
2348       \str_if_eq:VnTF \l_tmpa_str B {
2349         \__stex_notation_argument_assoc:n
2350       }{
2351         \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2352         \tl_put_right:Nx \l_stex_notation_dummyargs_tl {
2353           { \_stex_term_math_arg:nnn
2354             { \int_use:N \l__stex_notation_currarg_int }
2355             { \l_tmpa_str }
2356             { ###\int_use:N \l__stex_notation_currarg_int }
2357           }
2358         }
2359         \__stex_notation_arguments:
2360       }
2361     }
2362   }
2363 }

```

(End definition for `__stex_notation_arguments:.`)

`__stex_notation_argument_assoc:n`

```

2364 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2365
2366   \cs_generate_from_arg_count:NNnn \l_tmpa_cs \cs_set:Npn
2367     {\l__stex_notation_arity_str}{
2368     #1
2369   }
2370   \int_zero:N \l_tmpa_int
2371   \tl_clear:N \l_tmpa_tl
2372   \str_map_inline:Nn \l__stex_notation_args_str {
2373     \int_incr:N \l_tmpa_int
2374     \tl_put_right:Nx \l_tmpa_tl {
2375       \str_if_eq:nnTF {##1}{a}{ {} }{
2376         \str_if_eq:nnTF {##1}{B}{ {} }{
2377           {\_stex_term_arg:nn{\int_use:N \l_tmpa_int}{##### \int_use:N \l_tmpa_in
2378         }
2379       }
2380     }
2381   }
2382   \exp_after:wN\exp_after:wN\exp_after:wN \def
2383   \exp_after:wN\exp_after:wN\exp_after:wN \l_tmpa_cs
2384   \exp_after:wN\exp_after:wN\exp_after:wN ##
2385   \exp_after:wN\exp_after:wN\exp_after:wN 1
2386   \exp_after:wN\exp_after:wN\exp_after:wN ##

```

```

2387 \exp_after:wN\exp_after:wN\exp_after:wN 2
2388 \exp_after:wN\exp_after:wN\exp_after:wN {
2389   \exp_after:wN \exp_after:wN \exp_after:wN
2390   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN {
2391     \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2392   }
2393 }
2394
2395 \seq_pop_left:NN \l__stex_notation_remaining_precs_seq \l_tmpa_str
2396 \tl_put_right:Nx \l_stex_notation_dummyargs_tl { {
2397   \stex_term_math_assoc_arg:nnnn
2398   { \int_use:N \l__stex_notation_currarg_int }
2399   { \l_tmpa_str }
2400   { ####\int_use:N \l__stex_notation_currarg_int }
2401   { \l_tmpa_cs {####1} {####2} }
2402 } }
2403 \__stex_notation_arguments:
2404 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2405 \cs_new_protected:Nn \__stex_notation_final: {
2406 % \exp_args:Nne \use:nn
2407 % {
2408 % \cs_generate_from_arg_count:cNnn {
2409 %   stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2410 %   \l__stex_notation_suffix_str
2411 %   _cs
2412 % }
2413 % \cs_set:Npn \l__stex_notation_arity_str } { {
2414 %   \exp_after:wN \exp_after:wN \exp_after:wN
2415 %   \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2416 %   { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2417 % } }
2418
2419 % \tl_if_empty:NF \l__stex_notation_op_tl {
2420 %   \cs_set:cpx {
2421 %     stex_op_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2422 %     \l__stex_notation_suffix_str
2423 %     _cs
2424 %   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2425 % }
2426
2427 \exp_args:Nx \stex_do_up_to_module:n {
2428   \cs_generate_from_arg_count:cNnn {
2429     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2430     \l__stex_notation_suffix_str
2431     _cs
2432   } \cs_set:Npn { \l__stex_notation_arity_str } {
2433     \exp_after:wN \exp_after:wN \exp_after:wN
2434     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2435     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2436   }

```

```

2437 \tl_if_empty:NF \l__stex_notation_op_tl {
2438   \cs_set:cpn {
2439     stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2440     \l__stex_notation_suffix_str
2441     _cs
2442   } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2443 }
2444 }
2445
2446 \exp_args:Ne
2447 \stex_add_to_current_module:n {
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   } \cs_set:Npn {\l__stex_notation_arity_str} {
2453     \exp_after:wN \exp_after:wN \exp_after:wN
2454     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2455     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_sy
2456   }
2457   \tl_if_empty:NF \l__stex_notation_op_tl {
2458     \cs_set:cpn {
2459       stex_op_notation_\l_stex_get_symbol_uri_str \c_hash_str
2460       \l__stex_notation_suffix_str
2461       _cs
2462     } { \exp_not:N \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2463   }
2464 }
2465
2466 \stex_debug:nn{symbols}{
2467   Notation~\l__stex_notation_suffix_str
2468   ~for~\l_stex_get_symbol_uri_str^^J
2469   Operator~precedence:~\l__stex_notation_opprec_tl^^J
2470   Argument~precedences:~
2471   \seq_use:Nn \l__stex_notation_precedences_seq {,~}^^J
2472   Notation: \cs_meaning:c {
2473     stex_notation_ \l_stex_get_symbol_uri_str \c_hash_str
2474     \l__stex_notation_suffix_str
2475     _cs
2476   }
2477 }
2478
2479 \exp_args:Nx
2480 \stex_do_up_to_module:n {
2481   \seq_put_right:cx {
2482     l_stex_symdecl_ \l_stex_get_symbol_uri_str
2483     _notations
2484   } {
2485     \l__stex_notation_suffix_str
2486   }
2487 }
2488 \exp_args:Ne
2489 \stex_add_to_current_module:n {
2490   \seq_put_right:cn {

```



```

2491     \l_stex_symdecl \l_stex_get_symbol_uri_str
2492     _notations
2493   } { \l__stex_notation_suffix_str }
2494 }
2495
2496 \stex_if_smsmode:F {
2497
2498   % HTML annotations
2499   \stex_if_do_html:T {
2500     \stex_annotate_invisible:nnn { notation }
2501     { \l_stex_get_symbol_uri_str } {
2502       \stex_annotate_invisible:nnn { notationfragment }
2503       { \l__stex_notation_suffix_str }{}
2504       \stex_annotate_invisible:nnn { precedence }
2505       { \l__stex_notation_prec_str }{}
2506
2507       \int_zero:N \l_tmpa_int
2508       \str_set_eq:NN \l__stex_notation_remaining_args_str \l__stex_notation_args_str
2509       \tl_clear:N \l_tmpa_tl
2510       \int_step_inline:nn { \l__stex_notation_arity_str }{
2511         \int_incr:N \l_tmpa_int
2512         \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_notation_remaining_args_str }
2513         \str_set:Nx \l__stex_notation_remaining_args_str { \str_tail:N \l__stex_notation_r
2514         \str_if_eq:VnTF \l_tmpb_str a {
2515           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2516             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2517             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2518           } }
2519         }{
2520           \str_if_eq:VnTF \l_tmpb_str B {
2521             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2522               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2523               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2524             } }
2525           }{
2526             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2527               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2528             } }
2529           }
2530         }
2531       }
2532       \stex_annotate_invisible:nnn { notationcomp }{}{
2533         \str_set:Nx \l_stex_current_symbol_str { \l_stex_get_symbol_uri_str }
2534         $ \exp_args:Nno \use:nn { \use:c {
2535           stex_notation_ \l_stex_current_symbol_str
2536           \c_hash_str \l__stex_notation_suffix_str _cs
2537         } } { \l_tmpa_tl } $
2538       }
2539     }
2540   }
2541 }
2542 }

```

(End definition for _stex_notation_final:.)

`\setnotation`

```

2543 \keys_define:nn { stex / setnotation } {
2544   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2545   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2546   unknown .code:n = \str_set:Nx
2547     \l__stex_notation_variant_str \l_keys_key_str
2548 }
2549
2550 \cs_new_protected:Nn \stex_setnotation_args:n {
2551   \str_clear:N \l__stex_notation_lang_str
2552   \str_clear:N \l__stex_notation_variant_str
2553   \keys_set:nn { stex / setnotation } { #1 }
2554 }
2555
2556 \cs_new_protected:Nn \stex_setnotation:n {
2557   \exp_args:Nnx \seq_if_in:cnTF { l_stex_symdecl_#1 _notations }
2558     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{
2559     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2560       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2561     \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2562       { \c_hash_str }
2563     \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2564       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2565     \exp_args:Nx \stex_add_to_current_module:n {
2566       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2567         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2568       \exp_args:Nnx \seq_remove_all:cn { l_stex_symdecl_#1 _notations }
2569         { \c_hash_str }
2570       \exp_args:Nnx \seq_put_left:cn { l_stex_symdecl_#1 _notations }
2571         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2572     }
2573   \stex_debug:nn {notations}{
2574     Setting~default~notation~
2575     {\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str}~for~
2576     #1 \\
2577     \expandafter\meaning\csname
2578       l_stex_symdecl_#1 _notations\endcsname
2579   }
2580   }{
2581     % todo throw error
2582   }
2583 }
2584
2585 \NewDocumentCommand \setnotation {m m} {
2586   \stex_get_symbol:n { #1 }
2587   \stex_setnotation_args:n { #2 }
2588   \stex_setnotation:n{\l_stex_get_symbol_uri_str}
2589   \stex_smsmode_do:\ignorespacesandpars
2590 }
2591
2592 \cs_new_protected:Nn \stex_copy_notations:nn {
2593   \stex_debug:nn {notations}{
2594     Copying~notations~from~#2~to~#1\\
2595     \seq_use:cn{l_stex_symdecl_#2_notations}{,~}

```

```

2596 }
2597 \tl_clear:N \l_tmpa_tl
2598 \int_step_inline:nn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } } {
2599   \tl_put_right:Nn \l_tmpa_tl { {## ##1} }
2600 }
2601 \seq_map_inline:cn {l_stex_symdecl_#2_notations}{
2602   \cs_set_eq:Nc \l_tmpa_cs { stex_notation_ #2 \c_hash_str ##1 _cs }
2603   \edef \l_tmpa_tl {
2604     \exp_after:wN\exp_after:wN\exp_after:wN \exp_not:n
2605     \exp_after:wN\exp_after:wN\exp_after:wN {
2606       \exp_after:wN \l_tmpa_cs \l_tmpa_tl
2607     }
2608   }
2609   \exp_args:Nx
2610   \stex_do_up_to_module:n {
2611     \seq_put_right:cn{l_stex_symdecl_#1_notations}{##1}
2612     \cs_generate_from_arg_count:cNnn {
2613       stex_notation_ #1 \c_hash_str ##1 _cs
2614     } \cs_set:Npn { \prop_item:cn {l_stex_symdecl_#2_prop}{ arity } }{
2615       \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpa_tl}
2616     }
2617   }
2618 }
2619 }
2620
2621 \NewDocumentCommand \copynotation {m m} {
2622   \stex_get_symbol:n { #1 }
2623   \str_set_eq:NN \l_tmpa_str \l_stex_get_symbol_uri_str
2624   \stex_get_symbol:n { #2 }
2625   \exp_args:Noo
2626   \stex_copy_notations:nn \l_tmpa_str \l_stex_get_symbol_uri_str
2627   \exp_args:Nx \stex_add_import_to_current_module:n{
2628     \stex_copy_notations:nn {\l_tmpa_str} {\l_stex_get_symbol_uri_str}
2629   }
2630   \stex_smsmode_do:\ignorespacesandpars
2631 }
2632

```

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

\symdef

```

2633 \keys_define:nn { stex / symdef } {
2634   name .str_set_x:N = \l_stex_symdecl_name_str ,
2635   local .bool_set:N = \l_stex_symdecl_local_bool ,
2636   args .str_set_x:N = \l_stex_symdecl_args_str ,
2637   type .tl_set:N = \l_stex_symdecl_type_tl ,
2638   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2639   op .tl_set:N = \l_stex_notation_op_tl ,
2640   lang .str_set_x:N = \l_stex_notation_lang_str ,
2641   variant .str_set_x:N = \l_stex_notation_variant_str ,
2642   prec .str_set_x:N = \l_stex_notation_prec_str ,
2643   assoc .choices:nn =
2644     {bin,binl,binr,pre,conj,pwconj}
2645     {\str_set:Nx \l_stex_symdecl_assoctype_str {\l_keys_choice_tl}},

```

```

2646   unknown .code:n      = \str_set:Nx
2647     \l__stex_notation_variant_str \l_keys_key_str
2648 }
2649
2650 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2651   \str_clear:N \l_stex_symdecl_name_str
2652   \str_clear:N \l_stex_symdecl_args_str
2653   \str_clear:N \l_stex_symdecl_assoc_type_str
2654   \bool_set_false:N \l_stex_symdecl_local_bool
2655   \tl_clear:N \l_stex_symdecl_type_tl
2656   \tl_clear:N \l_stex_symdecl_definiens_tl
2657   \str_clear:N \l__stex_notation_lang_str
2658   \str_clear:N \l__stex_notation_variant_str
2659   \str_clear:N \l__stex_notation_prec_str
2660   \tl_clear:N \l__stex_notation_op_tl
2661
2662   \keys_set:nn { stex / symdef } { #1 }
2663 }
2664
2665 \NewDocumentCommand \symdef { m O{} } {
2666   \__stex_notation_symdef_args:n { #2 }
2667   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2668   \stex_symdecl_do:n { #1 }
2669   \tl_set:Nn \l_stex_notation_after_do_tl {
2670     \__stex_notation_final:
2671     \stex_smsmode_do:\ignorespacesandpars
2672   }
2673   \str_set:Nx \l_stex_get_symbol_uri_str {
2674     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2675   }
2676   \exp_args:Nx \stex_notation_do:nnnnn
2677     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { args } }
2678     { \prop_item:cn { \l_stex_symdecl \l_stex_get_symbol_uri_str _prop } { arity } }
2679     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2680     { \l__stex_notation_prec_str }
2681 }
2682 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

29.3 Variables

```

2683 <@@=stex_variables>
2684
2685 \keys_define:nn { stex / vardef } {
2686   name .str_set_x:N = \l__stex_variables_name_str ,
2687   args .str_set_x:N = \l__stex_variables_args_str ,
2688   type .tl_set:N    = \l__stex_variables_type_tl ,
2689   def .tl_set:N     = \l__stex_variables_def_tl ,
2690   op .tl_set:N      = \l__stex_variables_op_tl ,
2691   prec .str_set_x:N = \l__stex_variables_prec_str ,
2692   assoc .choices:nn =
2693     {bin,binl,binr,pre,conj,pwconj}
2694     {\str_set:Nx \l__stex_variables_assoc_type_str {\l_keys_choice_tl}},

```

```

2695     bind      .choices:nn      =
2696     {forall,exists}
2697     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2698   }
2699
2700   \cs_new_protected:Nn \__stex_variables_args:n {
2701     \str_clear:N \l__stex_variables_name_str
2702     \str_clear:N \l__stex_variables_args_str
2703     \str_clear:N \l__stex_variables_prec_str
2704     \str_clear:N \l__stex_variables_assoctype_str
2705     \str_clear:N \l__stex_variables_bind_str
2706     \tl_clear:N \l__stex_variables_type_tl
2707     \tl_clear:N \l__stex_variables_def_tl
2708     \tl_clear:N \l__stex_variables_op_tl
2709
2710     \keys_set:nn { stex / vardef } { #1 }
2711   }
2712
2713   \NewDocumentCommand \__stex_variables_do_simple:nnn { m O{} } {
2714     \__stex_variables_args:n {#2}
2715     \str_if_empty:NT \l__stex_variables_name_str {
2716       \str_set:Nx \l__stex_variables_name_str { #1 }
2717     }
2718     \prop_clear:N \l_tmpa_prop
2719     \prop_put:Nno \l_tmpa_prop { name } \l__stex_variables_name_str
2720
2721     \int_zero:N \l_tmpb_int
2722     \bool_set_true:N \l_tmpa_bool
2723     \str_map_inline:Nn \l__stex_variables_args_str {
2724       \token_case_meaning:NnF ##1 {
2725         0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
2726         {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
2727         {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
2728         {\tl_to_str:n a} {
2729           \bool_set_false:N \l_tmpa_bool
2730           \int_incr:N \l_tmpb_int
2731         }
2732         {\tl_to_str:n B} {
2733           \bool_set_false:N \l_tmpa_bool
2734           \int_incr:N \l_tmpb_int
2735         }
2736       }{
2737         \msg_error:nnxx{stex}{error/wrongargs}{
2738           variable~\l__stex_variables_name_str
2739         }{##1}
2740       }
2741     }
2742     \bool_if:NTF \l_tmpa_bool {
2743       % possibly numeric
2744       \str_if_empty:NTF \l__stex_variables_args_str {
2745         \prop_put:Nnn \l_tmpa_prop { args } {}
2746         \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
2747       }{
2748         \int_set:Nn \l_tmpa_int { \l__stex_variables_args_str }

```

```

2749     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
2750     \str_clear:N \l_tmpa_str
2751     \int_step_inline:nn \l_tmpa_int {
2752       \str_put_right:Nn \l_tmpa_str i
2753     }
2754     \str_set_eq:NN \l__stex_variables_args_str \l_tmpa_str
2755     \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2756   }
2757 } {
2758   \prop_put:Nnx \l_tmpa_prop { args } { \l__stex_variables_args_str }
2759   \prop_put:Nnx \l_tmpa_prop { arity }
2760   { \str_count:N \l__stex_variables_args_str }
2761 }
2762 \prop_put:Nnx \l_tmpa_prop { assoc } { \int_use:N \l_tmpb_int }
2763 \tl_set:cx { #1 } { \stex_invoke_variable:n { \l__stex_variables_name_str } }
2764
2765 \prop_set_eq:cN { l_stex_variable_\l__stex_variables_name_str _prop } \l_tmpa_prop
2766
2767 \tl_if_empty:NF \l__stex_variables_op_tl {
2768   \cs_set:cpx {
2769     stex_var_op_notation_\l__stex_variables_name_str_cs
2770   } { \exp_not:N\comp{ \exp_args:No \exp_not:n { \l__stex_variables_op_tl } } }
2771 }
2772
2773 \tl_set:Nn \l_stex_notation_after_do_tl {
2774   \exp_args:Nne \use:nn {
2775     \cs_generate_from_arg_count:cNnn { stex_var_notation_\l__stex_variables_name_str_cs }
2776     \cs_set:Npn { \prop_item:Nn \l_tmpa_prop { arity } }
2777   } {{
2778     \exp_after:wN \exp_after:wN \exp_after:wN
2779     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2780     { \exp_after:wN \l_stex_notation_macrocode_cs \l_stex_notation_dummyargs_tl \stex_symbol
2781   }}
2782 \stex_if_do_html:T {
2783   \stex_annotate_invisible:nnn {vardecl}{\l__stex_variables_name_str}{
2784     \stex_annotate_invisible:nnn { precedence }
2785     { \l__stex_variables_prec_str }{}
2786   \tl_if_empty:NF \l__stex_variables_type_tl {\stex_annotate_invisible:nnn{type}{\l__stex_variables_type_str}{
2787     \stex_annotate_invisible:nnn{args}{\l__stex_variables_args_str }
2788     \stex_annotate_invisible:nnn{macroname}{#1}{
2789   \tl_if_empty:NF \l__stex_variables_def_tl {
2790     \stex_annotate_invisible:nnn{definiens}{
2791       {\l__stex_variables_def_tl$}
2792     }
2793   \str_if_empty:NF \l__stex_variables_assoc_type_str {
2794     \stex_annotate_invisible:nnn{assoc_type}{\l__stex_variables_assoc_type_str}{
2795   }
2796   \int_zero:N \l_tmpa_int
2797   \str_set_eq:NN \l__stex_variables_remaining_args_str \l__stex_variables_args_str
2798   \tl_clear:N \l_tmpa_tl
2799   \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } } {{
2800     \int_incr:N \l_tmpa_int
2801     \str_set:Nx \l_tmpb_str { \str_head:N \l__stex_variables_remaining_args_str }
2802     \str_set:Nx \l__stex_variables_remaining_args_str { \str_tail:N \l__stex_variables

```

```

2803 \str_if_eq:VnTF \l_tmpb_str a {
2804 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2805 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2806 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2807 } }
2808 }{
2809 \str_if_eq:VnTF \l_tmpb_str B {
2810 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2811 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2812 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2813 } }
2814 }{
2815 \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2816 \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2817 } }
2818 }
2819 }
2820 }
2821 \stex_annotate_invisible:nnn { notationcomp }{}{
2822 \str_set:Nx \l_stex_current_symbol_str {var://\l__stex_variables_name_str }
2823 $ \exp_args:Nno \use:nn { \use:c {
2824 stex_var_notation_\l__stex_variables_name_str _cs
2825 } } { \l_tmpa_tl } $
2826 }
2827 }
2828 }\ignorespacesandpars
2829 }
2830
2831 \stex_notation_do:nnnn { \l__stex_variables_args_str } { \prop_item:Nn \l_tmpa_prop { ari
2832 }
2833
2834 \cs_new:Nn \_stex_reset:N {
2835 \tl_if_exist:NTF #1 {
2836 \def \exp_not:N #1 { \exp_args:No \exp_not:n #1 }
2837 }{
2838 \let \exp_not:N #1 \exp_not:N \undefined
2839 }
2840 }
2841
2842 \NewDocumentCommand \__stex_variables_do_complex:nn { m m }{
2843 \clist_set:Nx \l__stex_variables_names { \tl_to_str:n {#1} }
2844 \exp_args:Nnx \use:nn {
2845 % TODO
2846 \stex_annotate_invisible:nnn {vardecls}{\clist_use:Nn\l__stex_variables_names,}{
2847 #2
2848 }
2849 }{
2850 \_stex_reset:N \varnot
2851 \_stex_reset:N \vartype
2852 \_stex_reset:N \vardefi
2853 }
2854 }
2855
2856 \NewDocumentCommand \vardef { s } {

```

```

2857 \IfBooleanTF#1 {
2858   \__stex_variables_do_complex:nn
2859 }{
2860   \__stex_variables_do_simple:nnn
2861 }
2862 }
2863
2864 \NewDocumentCommand \svar { 0{} m }{
2865   \tl_if_empty:nTF {#1}{
2866     \str_set:Nn \l_tmpa_str { #2 }
2867   }{
2868     \str_set:Nn \l_tmpa_str { #1 }
2869   }
2870   \_stex_term_omv:nn {
2871     var://\l_tmpa_str
2872   }{
2873     \exp_args:Nnx \use:nn {
2874       \def\comp{\_varcomp}
2875       \str_set:Nx \l_stex_current_symbol_str { var://\l_tmpa_str }
2876       \comp{ #2 }
2877     }{
2878       \_stex_reset:N \comp
2879       \_stex_reset:N \l_stex_current_symbol_str
2880     }
2881   }
2882 }
2883
2884
2885
2886 \keys_define:nn { stex / varseq } {
2887   name .str_set_x:N = \l__stex_variables_name_str ,
2888   args .int_set:N   = \l__stex_variables_args_int ,
2889   type .tl_set:N    = \l__stex_variables_type_tl ,
2890   mid .tl_set:N     = \l__stex_variables_mid_tl ,
2891   bind .choices:nn =
2892     {forall,exists}
2893     {\str_set:Nx \l__stex_variables_bind_str {\l_keys_choice_tl}}
2894 }
2895
2896 \cs_new_protected:Nn \__stex_variables_seq_args:n {
2897   \str_clear:N \l__stex_variables_name_str
2898   \int_set:Nn \l__stex_variables_args_int 1
2899   \tl_clear:N \l__stex_variables_type_tl
2900   \str_clear:N \l__stex_variables_bind_str
2901
2902   \keys_set:nn { stex / varseq } { #1 }
2903 }
2904
2905 \NewDocumentCommand \varseq {m 0{} m m m}{
2906   \__stex_variables_seq_args:n { #2 }
2907   \str_if_empty:NT \l__stex_variables_name_str {
2908     \str_set:Nx \l__stex_variables_name_str { #1 }
2909   }
2910   \prop_clear:N \l_tmpa_prop

```



```

2911 \prop_put:Nnx \l_tmpa_prop { arity }{\int_use:N \l__stex_variables_args_int}
2912
2913 \seq_set_from_clist:Nn \l_tmpa_seq {#3}
2914 \int_compare:nNnF {\seq_count:N \l_tmpa_seq} = \l__stex_variables_args_int {
2915   \msg_error:nnxx{stex}{error/seqlength}
2916   {\int_use:N \l__stex_variables_args_int}
2917   {\seq_count:N \l_tmpa_seq}
2918 }
2919 \seq_set_from_clist:Nn \l_tmpb_seq {#4}
2920 \int_compare:nNnF {\seq_count:N \l_tmpb_seq} = \l__stex_variables_args_int {
2921   \msg_error:nnxx{stex}{error/seqlength}
2922   {\int_use:N \l__stex_variables_args_int}
2923   {\seq_count:N \l_tmpb_seq}
2924 }
2925 \prop_put:Nnn \l_tmpa_prop {starts} {#3}
2926 \prop_put:Nnn \l_tmpa_prop {ends} {#4}
2927
2928 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2929   \cs_set:Npn { \int_use:N \l__stex_variables_args_int } { #5 }
2930
2931 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2932 \int_step_inline:nn \l__stex_variables_args_int {
2933   \tl_put_right:Nx \l_tmpa_tl { {\seq_item:Nn \l_tmpa_seq {##1}} }
2934 }
2935 \tl_set:Nx \l_tmpa_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpa_tl}}
2936 \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2937 \tl_if_empty:NF \l__stex_variables_mid_tl {
2938   \tl_put_right:No \l_tmpa_tl \l__stex_variables_mid_tl
2939   \tl_put_right:Nn \l_tmpa_tl {,\ellipses,}
2940 }
2941 \exp_args:NNo \tl_set:No \l_tmpb_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2942 \int_step_inline:nn \l__stex_variables_args_int {
2943   \tl_put_right:Nx \l_tmpb_tl { {\seq_item:Nn \l_tmpb_seq {##1}} }
2944 }
2945 \tl_set:Nx \l_tmpb_tl {\exp_args:NNo \exp_args:No \exp_not:n{\l_tmpb_tl}}
2946 \tl_put_right:No \l_tmpa_tl \l_tmpb_tl
2947
2948
2949 \prop_put:Nno \l_tmpa_prop { notation }\l_tmpa_tl
2950
2951 \tl_set:cx {#1} {\stex_invoke_sequence:n {\l__stex_variables_name_str}}
2952
2953 \exp_args:NNo \tl_set:No \l_tmpa_tl {\use:c{stex_varseq_\l__stex_variables_name_str _cs}}
2954
2955 \int_step_inline:nn \l__stex_variables_args_int {
2956   \tl_set:Nx \l_tmpa_tl {\exp_args:No \exp_not:n \l_tmpa_tl {
2957     \stex_term_math_arg:nnn{##1}{0}{\exp_not:n{####}##1}
2958   }}
2959 }
2960
2961 \tl_set:Nx \l_tmpa_tl {
2962   \stex_term_math_oma:nnnn { varseq://\l__stex_variables_name_str }{}{0}{
2963     \exp_args:NNo \exp_args:No \exp_not:n {\l_tmpa_tl}
2964   }

```

```

2965 }
2966
2967 \tl_set:No \l_tmpa_tl { \exp_after:wN { \l_tmpa_tl \stex_symbol_after_invokation_tl} }
2968
2969 \exp_args:Nno \use:nn {
2970 \cs_generate_from_arg_count:cNnn {stex_varseq_\l__stex_variables_name_str _cs}
2971 \cs_set:Npn {\int_use:N \l__stex_variables_args_int}}{\l_tmpa_tl}
2972
2973 \stex_debug:nn{sequences}{New~Sequence:~
2974 \expandafter\meaning\csname stex_varseq_\l__stex_variables_name_str _cs\endcsname\\~\\
2975 \prop_to_keyval:N \l_tmpa_prop
2976 }
2977
2978 \prop_set_eq:cN {stex_varseq_\l__stex_variables_name_str _prop}\l_tmpa_prop
2979 \ignorespacesandpars
2980 }
2981
2982 </package>

```

Chapter 30

STEX -Terms Implementation

```
2983 <*package>
2984
2985 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2986
2987 <@@=stex_terms>
2988
2989 Warnings and error messages
2990 \msg_new:nnn{stex}{error/nonotation}{
2991   Symbol~#1~invoked,~but~has~no~notation~#2!
2992 }
2993 \msg_new:nnn{stex}{error/notationarg}{
2994   Error~in~parsing~notation~#1
2995 }
2996 \msg_new:nnn{stex}{error/noop}{
2997   Symbol~#1~has~no~operator~notation~for~notation~#2
2998 }
2999 \msg_new:nnn{stex}{error/notallowed}{
3000   Symbol~invocation~#1~not~allowed~in~notation~component~of~#2
3001 }
```

30.1 Symbol Invocations

`\stex_invoke_symbol:n` Invokes a semantic macro

```
3001
3002
3003 \bool_new:N \l_stex_allow_semantic_bool
3004 \bool_set_true:N \l_stex_allow_semantic_bool
3005
3006 \cs_new_protected:Nn \stex_invoke_symbol:n {
3007   \bool_if:NTF \l_stex_allow_semantic_bool {
3008     \str_if_eq:eeF {
3009       \prop_item:cn {
3010         l_stex_symdecl_#1_prop
3011       }{ deprecate }
3012     }
```

```

3012   }{}{
3013     \msg_warning:nnxx{stex}{warning/deprecated}{
3014       Symbol~#1
3015     }{
3016       \prop_item:cn {l_stex_symdecl_#1_prop}{ deprecate }
3017     }
3018   }
3019   \if_mode_math:
3020     \exp_after:wN \__stex_terms_invoke_math:n
3021   \else:
3022     \exp_after:wN \__stex_terms_invoke_text:n
3023   \fi: { #1 }
3024 }{
3025   \msg_error:nnxx{stex}{error/notallowed}{#1}{\l_stex_current_symbol_str}
3026 }
3027 }
3028
3029 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
3030   \peek_charcode_remove:NTF ! {
3031     \__stex_terms_invoke_op_custom:nn {#1}
3032   }{
3033     \__stex_terms_invoke_custom:nn {#1}
3034   }
3035 }
3036
3037 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
3038   \peek_charcode_remove:NTF ! {
3039     % operator
3040     \peek_charcode_remove:NTF * {
3041       % custom op
3042       \__stex_terms_invoke_op_custom:nn {#1}
3043     }{
3044       % op notation
3045       \peek_charcode:NTF [ {
3046         \__stex_terms_invoke_op_notation:nw {#1}
3047       }{
3048         \__stex_terms_invoke_op_notation:nw {#1}[]
3049       }
3050     }
3051   }{
3052     \peek_charcode_remove:NTF * {
3053       \__stex_terms_invoke_custom:nn {#1}
3054       % custom
3055     }{
3056       % normal
3057       \peek_charcode:NTF [ {
3058         \__stex_terms_invoke_notation:nw {#1}
3059       }{
3060         \__stex_terms_invoke_notation:nw {#1}[]
3061       }
3062     }
3063   }
3064 }
3065

```

```

3066
3067 \cs_new_protected:Nn \__stex_terms_invoke_op_custom:nn {
3068   \exp_args:Nnx \use:nn {
3069     \def\comp{\_comp}
3070     \str_set:Nn \l_stex_current_symbol_str { #1 }
3071     \bool_set_false:N \l_stex_allow_semantic_bool
3072     \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
3073       \comp{ #2 }
3074     }
3075   }{
3076     \stex_reset:N \comp
3077     \stex_reset:N \l_stex_current_symbol_str
3078     \bool_set_true:N \l_stex_allow_semantic_bool
3079   }
3080 }
3081
3082 \keys_define:nn { stex / terms } {
3083   lang .tl_set_x:N = \l_stex_notation_lang_str ,
3084   variant .tl_set_x:N = \l_stex_notation_variant_str ,
3085   unknown .code:n = \str_set:Nx
3086     \l_stex_notation_variant_str \l_keys_key_str
3087 }
3088
3089 \cs_new_protected:Nn \__stex_terms_args:n {
3090   \str_clear:N \l_stex_notation_lang_str
3091   \str_clear:N \l_stex_notation_variant_str
3092
3093   \keys_set:nn { stex / terms } { #1 }
3094 }
3095
3096 \cs_new_protected:Nn \stex_find_notation:nn {
3097   \__stex_terms_args:n { #2 }
3098   \seq_if_empty:cTF {
3099     l_stex_symdecl_ #1 _notations
3100   } {
3101     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
3102   } {
3103     \bool_lazy_all:nTF {
3104       {\str_if_empty_p:N \l_stex_notation_variant_str}
3105       {\str_if_empty_p:N \l_stex_notation_lang_str}
3106     }{
3107       \seq_get_left:cN {l_stex_symdecl_#1_notations}\l_stex_notation_variant_str
3108     }{
3109       \seq_if_in:cxTF {l_stex_symdecl_#1_notations}{
3110         \l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3111       }{
3112         \str_set:Nx \l_stex_notation_variant_str { \l_stex_notation_variant_str \c_hash_str
3113       }{
3114         \msg_error:nnxx{stex}{error/nonotation}{#1}{
3115           ~\l_stex_notation_variant_str \c_hash_str \l_stex_notation_lang_str
3116         }
3117       }
3118     }
3119   }

```

```

3120 }
3121
3122 \cs_new_protected:Npn \__stex_terms_invoke_op_notation:nw #1 [#2] {
3123   \exp_args:Nnx \use:nn {
3124     \def\comp{\_comp}
3125     \str_set:Nn \l_stex_current_symbol_str { #1 }
3126     \stex_find_notation:nn { #1 }{ #2 }
3127     \bool_set_false:N \l_stex_allow_semantic_bool
3128     \cs_if_exist:cTF {
3129       stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3130     }{
3131       \_stex_term_oms:nnn {
3132         #1 \c_hash_str \l_stex_notation_variant_str
3133       }{ #1 }{
3134         \use:c{stex_op_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3135       }
3136     }{
3137       \int_compare:nNnTF {\prop_item:cn {l_stex_symdecl_#1_prop}{arity}} = 0{
3138         \cs_if_exist:cTF {
3139           stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3140         }{
3141           \tl_set:Nx \stex_symbol_after_invokation_tl {
3142             \_stex_reset:N \comp
3143             \_stex_reset:N \stex_symbol_after_invokation_tl
3144             \_stex_reset:N \l_stex_current_symbol_str
3145             \bool_set_true:N \l_stex_allow_semantic_bool
3146           }
3147           \def\comp{\_comp}
3148           \str_set:Nn \l_stex_current_symbol_str { #1 }
3149           \bool_set_false:N \l_stex_allow_semantic_bool
3150           \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3151         }{
3152           \msg_error:nnxx{stex}{error/nonotation}{#1}{
3153             ~\l_stex_notation_variant_str
3154           }
3155         }
3156       }{
3157         \msg_error:nnxx{stex}{error/noop}{#1}{\l_stex_notation_variant_str}
3158       }
3159     }
3160   }{
3161     \_stex_reset:N \comp
3162     \_stex_reset:N \l_stex_current_symbol_str
3163     \bool_set_true:N \l_stex_allow_semantic_bool
3164   }
3165 }
3166
3167 \cs_new_protected:Npn \__stex_terms_invoke_notation:nw #1 [#2] {
3168   \stex_find_notation:nn { #1 }{ #2 }
3169   \cs_if_exist:cTF {
3170     stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs
3171   }{
3172     \tl_set:Nx \stex_symbol_after_invokation_tl {
3173       \_stex_reset:N \comp

```

```

3174 \stex_reset:N \stex_symbol_after_invokation_tl
3175 \stex_reset:N \l_stex_current_symbol_str
3176 \bool_set_true:N \l_stex_allow_semantic_bool
3177 }
3178 \def\comp{\_comp}
3179 \str_set:Nn \l_stex_current_symbol_str { #1 }
3180 \bool_set_false:N \l_stex_allow_semantic_bool
3181 \use:c{stex_notation_ #1 \c_hash_str \l_stex_notation_variant_str _cs}
3182 }{
3183 \msg_error:nnxx{stex}{error/nonotation}{#1}{
3184 ~\l_stex_notation_variant_str
3185 }
3186 }
3187 }
3188
3189 \prop_new:N \l__stex_terms_custom_args_prop
3190
3191 \cs_new_protected:Nn \__stex_terms_invoke_custom:nn {
3192 \exp_args:Nnx \use:nn {
3193 \bool_set_false:N \l_stex_allow_semantic_bool
3194 \def\comp{\_comp}
3195 \str_set:Nn \l_stex_current_symbol_str { #1 }
3196 \prop_clear:N \l__stex_terms_custom_args_prop
3197 \prop_put:Nnn \l__stex_terms_custom_args_prop {currnum} {1}
3198 \prop_get:cnN {
3199 \l_stex_symdecl_#1 _prop
3200 }{ args } \l_tmpa_str
3201 \prop_put:Nno \l__stex_terms_custom_args_prop {args} \l_tmpa_str
3202 \tl_set:Nn \arg { \__stex_terms_arg: }
3203 \str_if_empty:NTF \l_tmpa_str {
3204 \stex_term_oms:nnn {#1}{#1}{#2}
3205 }{
3206 \str_if_in:NnTF \l_tmpa_str b {
3207 \stex_term_ombind:nnn {#1}{#1}{#2}
3208 }{
3209 \str_if_in:NnTF \l_tmpa_str B {
3210 \stex_term_ombind:nnn {#1}{#1}{#2}
3211 }{
3212 \stex_term_oma:nnn {#1}{#1}{#2}
3213 }
3214 }
3215 }
3216 % TODO check that all arguments exist
3217 }{
3218 \stex_reset:N \l_stex_current_symbol_str
3219 \stex_reset:N \arg
3220 \stex_reset:N \comp
3221 \stex_reset:N \l__stex_terms_custom_args_prop
3222 \bool_set_true:N \l_stex_allow_semantic_bool
3223 }
3224 }
3225
3226 \NewDocumentCommand \__stex_terms_arg: { s O{} m}{
3227 \tl_if_empty:nTF {#2}{

```

```

3228 \int_set:Nn \l_tmpa_int {\prop_item:Nn \l__stex_terms_custom_args_prop {currnum}}
3229 \bool_set_true:N \l_tmpa_bool
3230 \bool_do_while:Nn \l_tmpa_bool {
3231   \exp_args:NNx \prop_if_in:NnTF \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int}
3232   \int_incr:N \l_tmpa_int
3233   }{
3234     \bool_set_false:N \l_tmpa_bool
3235   }
3236 }
3237 }{
3238   \int_set:Nn \l_tmpa_int { #2 }
3239   \exp_args:NNx \prop_if_in:NnT \l__stex_terms_custom_args_prop {\int_use:N \l_tmpa_int} {
3240     % TODO throw error
3241   }
3242 }
3243 \str_set:Nx \l_tmpa_str {\prop_item:Nn \l__stex_terms_custom_args_prop {args} }
3244 \int_compare:nNnT \l_tmpa_int > {\str_count:N \l_tmpa_str} {
3245   % TODO throw error
3246 }
3247 \bool_set_true:N \l_stex_allow_semantic_bool
3248 \IfBooleanTF#1{
3249   \stex_annotate_invisible:n {
3250     \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3251   }
3252 }{
3253   \exp_args:No \_stex_term_arg:nn {\l_stex_current_symbol_str}{#3}
3254 }
3255 \bool_set_false:N \l_stex_allow_semantic_bool
3256 }
3257
3258
3259 \cs_new_protected:Nn \_stex_term_arg:nn {
3260   \bool_set_true:N \l_stex_allow_semantic_bool
3261   \stex_annotate:nnn{ arg }{ #1 }{ #2 }
3262   \bool_set_false:N \l_stex_allow_semantic_bool
3263 }
3264
3265 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
3266   \exp_args:Nnx \use:nn
3267   { \int_set:Nn \l__stex_terms_downprec { #2 }
3268     \_stex_term_arg:nn { #1 }{ #3 }
3269   }
3270   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3271   }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

3272 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
3273   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #4 }
3274   \tl_set:Nn \l_tmpb_tl {\_stex_term_math_arg:nnn{#1}{#2}}
3275   \exp_args:Nx \tl_if_empty:nTF { \tl_tail:n{ #3 }}{
3276     \expandafter\if\expandafter\relax\noexpand#3
3277     \expandafter\__stex_terms_math_assoc_arg_maybe_sequence:N\expandafter#3

```



```

3278 \else\expandafter\__stex_terms_math_assoc_arg_simple:n\expandafter#3\fi
3279 }{
3280 \__stex_terms_math_assoc_arg_simple:n{#3}
3281 }
3282 }
3283
3284 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_maybe_sequence:N {
3285 \str_set:Nx \l_tmpa_str { \cs_argument_spec:N #1 }
3286 \str_if_empty:NTF \l_tmpa_str {
3287 \exp_args:Nx \cs_if_eq:NNTF {
3288 \tl_head:N #1
3289 } \stex_invoke_sequence:n {
3290 \tl_set:Nx \l_tmpa_tl {\tl_tail:N #1}
3291 \str_set:Nx \l_tmpa_str {\exp_after:wN \use:n \l_tmpa_tl}
3292 \tl_set:Nx \l_tmpa_tl {\prop_item:cn {stex_varseq \l_tmpa_str _prop}{notation}}
3293 \exp_args:NNo \seq_set_from_clist:Nn \l_tmpa_seq \l_tmpa_tl
3294 \tl_set:Nx \l_tmpa_tl {\exp_not:N \exp_not:n{
3295 \exp_not:n{\exp_args:Nnx \use:nn} {
3296 \exp_not:n {
3297 \def\comp{\_varcomp}
3298 \str_set:Nn \l_stex_current_symbol_str
3299 } {varseq://\l_tmpa_str}
3300 \exp_not:n{ ##1 }
3301 }{
3302 \exp_not:n {
3303 \_stex_reset:N \comp
3304 \_stex_reset:N \l_stex_current_symbol_str
3305 }
3306 }
3307 }}}
3308 \exp_args:Nno \use:nn {\seq_set_map:NNn \l_tmpa_seq \l_tmpa_seq} \l_tmpa_tl
3309 \seq_reverse:N \l_tmpa_seq
3310 \seq_pop:NN \l_tmpa_seq \l_tmpa_tl
3311 \seq_map_inline:Nn \l_tmpa_seq {
3312 \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
3313 \exp_args:Nno
3314 \l_tmpa_cs { ##1 } \l_tmpa_tl
3315 }
3316 }
3317 \tl_set:Nx \l_tmpa_tl {
3318 \_stex_term_omv:nn {varseq://\l_tmpa_str}{
3319 \exp_args:No \exp_not:n \l_tmpa_tl
3320 }
3321 }
3322 \exp_args:No\l_tmpb_tl\l_tmpa_tl
3323 }{
3324 \__stex_terms_math_assoc_arg_simple:n { #1 }
3325 }
3326 } {
3327 \__stex_terms_math_assoc_arg_simple:n { #1 }
3328 }
3329
3330 }
3331

```

```

3332 \cs_new_protected:Nn \__stex_terms_math_assoc_arg_simple:n {
3333   \clist_set:Nn \l_tmpa_clist{ #1 }
3334   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
3335     \tl_set:Nn \l_tmpa_tl { #1 }
3336   }{
3337     \clist_reverse:N \l_tmpa_clist
3338     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
3339
3340     \clist_map_inline:Nn \l_tmpa_clist {
3341       \exp_args:NNNo \exp_args:NNNo \tl_set:No \l_tmpa_tl {
3342         \exp_args:Nno
3343         \l_tmpa_cs { ##1 } \l_tmpa_tl
3344       }
3345     }
3346   }
3347   \exp_args:No\l_tmpb_tl\l_tmpa_tl
3348 }

```

(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
3349 \tl_const:Nx \infprec {\int_use:N \c_max_int}
3350 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
3351 \int_new:N \l__stex_terms_downprec
3352 \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
3353 \tl_set:Nn \l__stex_terms_left_bracket_str (
3354 \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

```

3355 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
3356   \bool_if:NTF \l__stex_terms_brackets_done_bool {
3357     \bool_set_false:N \l__stex_terms_brackets_done_bool
3358     #2
3359   } {
3360     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
3361       \bool_if:NTF \l_stex_inarray_bool { #2 }{
3362         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
3363         \dobrackets { #2 }
3364       }
3365     }{ #2 }
3366   }
3367 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

3368 \bool_new:N \l__stex_terms_brackets_done_bool
3369 %\RequirePackage{scalerel}
3370 \cs_new_protected:Npn \dobrackets #1 {
3371   %\ThisStyle{\if D\m@switch
3372   %   \exp_args:Nnx \use:nn
3373   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
3374   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
3375   % \else
3376   \exp_args:Nnx \use:nn
3377   {
3378     \bool_set_true:N \l__stex_terms_brackets_done_bool
3379     \int_set:Nn \l__stex_terms_downprec \infpref
3380     \l__stex_terms_left_bracket_str
3381     #1
3382   }
3383   {
3384     \bool_set_false:N \l__stex_terms_brackets_done_bool
3385     \l__stex_terms_right_bracket_str
3386     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
3387   }
3388   %\fi}
3389 }

```

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

`\withbrackets`

```

3390 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
3391   \exp_args:Nnx \use:nn
3392   {
3393     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
3394     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
3395     #3
3396   }
3397   {
3398     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
3399     {\l__stex_terms_left_bracket_str}
3400     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
3401     {\l__stex_terms_right_bracket_str}
3402   }
3403 }

```

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

`\STEXinvisible`

```

3404 \cs_new_protected:Npn \STEXinvisible #1 {
3405   \stex_annotate_invisible:n { #1 }
3406 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
3407 \cs_new_protected:Nn \_stex_term_oms:nnn {
3408   \stex_annotate:nnn{ OMID }{ #2 }{
3409     \stex_highlight_term:nn { #1 } { #3 }
3410   }
3411 }
3412
3413 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
3414   \__stex_terms_maybe_brackets:nn { #3 }{
3415     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3416   }
3417 }
```

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

`_stex_term_math_omv:nn`

```
3418 \cs_new_protected:Nn \_stex_term_omv:nn {
3419   \stex_annotate:nnn{ OMV }{ #1 }{
3420     \stex_highlight_term:nn { #1 } { #2 }
3421   }
3422 }
```

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

`_stex_term_math_oma:nnnn`

```
3423 \cs_new_protected:Nn \_stex_term_oma:nnn {
3424   \stex_annotate:nnn{ OMA }{ #2 }{
3425     \stex_highlight_term:nn { #1 } { #3 }
3426   }
3427 }
3428
3429 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
3430   \__stex_terms_maybe_brackets:nn { #3 }{
3431     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3432   }
3433 }
```

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

`_stex_term_math_omb:nnnn`

```
3434 \cs_new_protected:Nn \_stex_term_ombind:nnn {
3435   \stex_annotate:nnn{ OMBIND }{ #2 }{
3436     \stex_highlight_term:nn { #1 } { #3 }
3437   }
3438 }
3439
3440 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
3441   \__stex_terms_maybe_brackets:nn { #3 }{
3442     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
3443   }
3444 }
```

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

```

\symref
\symname
3445 \cs_new:Nn \stex_capitalize:n { \uppercase{#1} }
3446
3447 \keys_define:nn { stex / symname } {
3448   pre      .tl_set_x:N      = \l__stex_terms_pre_tl ,
3449   post     .tl_set_x:N      = \l__stex_terms_post_tl ,
3450   root     .tl_set_x:N      = \l__stex_terms_root_tl
3451 }
3452
3453 \cs_new_protected:Nn \stex_symname_args:n {
3454   \tl_clear:N \l__stex_terms_post_tl
3455   \tl_clear:N \l__stex_terms_pre_tl
3456   \tl_clear:N \l__stex_terms_root_str
3457   \keys_set:nn { stex / symname } { #1 }
3458 }
3459
3460 \NewDocumentCommand \symref { m m }{
3461   \let\compemph_uri_prev:\compemph@uri
3462   \let\compemph@uri\symrefemph@uri
3463   \STEXsymbol{#1}!\{ #2 }
3464   \let\compemph@uri\compemph_uri_prev:
3465 }
3466
3467 \NewDocumentCommand \synonym { 0{} m m }{
3468   \stex_symname_args:n { #1 }
3469   \let\compemph_uri_prev:\compemph@uri
3470   \let\compemph@uri\symrefemph@uri
3471   % TODO
3472   \STEXsymbol{#2}!\{\l__stex_terms_pre_tl #3 \l__stex_terms_post_tl}
3473   \let\compemph@uri\compemph_uri_prev:
3474 }
3475
3476 \NewDocumentCommand \symname { 0{} m }{
3477   \stex_symname_args:n { #1 }
3478   \stex_get_symbol:n { #2 }
3479   \str_set:Nx \l_tmpa_str {
3480     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3481   }
3482   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3483
3484   \let\compemph_uri_prev:\compemph@uri
3485   \let\compemph@uri\symrefemph@uri
3486   \exp_args:NNx \use:nn
3487   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!\{
3488     \l__stex_terms_pre_tl \l_tmpa_str \l__stex_terms_post_tl
3489   } }
3490   \let\compemph@uri\compemph_uri_prev:
3491 }
3492
3493 \NewDocumentCommand \Symname { 0{} m }{
3494   \stex_symname_args:n { #1 }
3495   \stex_get_symbol:n { #2 }
3496   \str_set:Nx \l_tmpa_str {
3497     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }

```

```

3498 }
3499 \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
3500 \let\compemph_uri_prev:\compemph@uri
3501 \let\compemph@uri\symrefemph@uri
3502 \exp_args:NNx \use:nn
3503 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }!{
3504   \exp_after:wN \stex_capitalize:n \l_tmpa_str
3505   \l__stex_terms_post_tl
3506 } }
3507 \let\compemph@uri\compemph_uri_prev:
3508 }

```

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

30.3 Notation Components

```

3509 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

3510 \cs_new_protected:Nn \stex_highlight_term:nn {
3511   #2
3512 }
3513
3514 \cs_new_protected:Nn \stex_unhighlight_term:n {
3515   % \latexml_if:TF {
3516   %   #1
3517   % } {
3518   %   \rustex_if:TF {
3519   %     #1
3520   %   } {
3521   %     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
3522   %   }
3523   % }
3524 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
\varemp
\varemp@uri
3525 \cs_new_protected:Npn \_comp #1 {
3526   \str_if_empty:NF \l_stex_current_symbol_str {
3527     \rustex_if:TF {
3528       \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
3529     }{
3530       \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
3531     }
3532   }
3533 }
3534
3535 \cs_new_protected:Npn \_varcomp #1 {
3536   \str_if_empty:NF \l_stex_current_symbol_str {
3537     \rustex_if:TF {
3538       \stex_annotate:nnn { varcomp }{ \l_stex_current_symbol_str }{ #1 }
3539     }{
3540       \exp_args:Nnx \varemp@uri { #1 } { \l_stex_current_symbol_str }

```

```

3541     }
3542   }
3543 }
3544
3545 \def\comp{\_comp}
3546
3547 \cs_new_protected:Npn \compemph@uri #1 #2 {
3548   \compemph{ #1 }
3549 }
3550
3551
3552 \cs_new_protected:Npn \compemph #1 {
3553   #1
3554 }
3555
3556 \cs_new_protected:Npn \defemph@uri #1 #2 {
3557   \defemph{#1}
3558 }
3559
3560 \cs_new_protected:Npn \defemph #1 {
3561   \textbf{#1}
3562 }
3563
3564 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
3565   \symrefemph{#1}
3566 }
3567
3568 \cs_new_protected:Npn \symrefemph #1 {
3569   \textbf{#1}
3570 }
3571
3572 \cs_new_protected:Npn \varemp@uri #1 #2 {
3573   \varemp{#1}
3574 }
3575
3576 \cs_new_protected:Npn \varemp #1 {
3577   #1
3578 }

```

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

\ellipses

```

3579 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix 3580 \bool_new:N \l_stex_inparray_bool
\parrayline 3581 \bool_set_false:N \l_stex_inparray_bool
\parraylineh 3582 \NewDocumentCommand \parray { m m } {
\parraycell 3583   \begingroup
3584   \bool_set_true:N \l_stex_inparray_bool
3585   \begin{array}{#1}
3586     #2
3587   \end{array}

```

```

3588 \endgroup
3589 }
3590
3591 \NewDocumentCommand \prmatrix { m } {
3592   \begingroup
3593   \bool_set_true:N \l_stex_inarray_bool
3594   \begin{matrix}
3595     #1
3596   \end{matrix}
3597   \endgroup
3598 }
3599
3600 \def \maybepline {
3601   \bool_if:NT \l_stex_inarray_bool {\hline}
3602 }
3603
3604 \def \parrayline #1 #2 {
3605   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
3606 }
3607
3608 \def \pmrow #1 { \parrayline{}{ #1 } }
3609
3610 \def \parraylineh #1 #2 {
3611   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
3612 }
3613
3614 \def \parraycell #1 {
3615   #1 \bool_if:NT \l_stex_inarray_bool {&}
3616 }

```

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

30.4 Variables

```

3617 <@@=stex_variables>

```

\stex_invoke_variable:n Invokes a variable

```

3618 \cs_new_protected:Nn \stex_invoke_variable:n {
3619   \if_mode_math:
3620     \exp_after:wN \__stex_variables_invoke_math:n
3621   \else:
3622     \exp_after:wN \__stex_variables_invoke_text:n
3623   \fi: {#1}
3624 }
3625
3626 \cs_new_protected:Nn \__stex_variables_invoke_text:n {
3627   %TODO
3628 }
3629
3630
3631 \cs_new_protected:Nn \__stex_variables_invoke_math:n {
3632   \peek_charcode_remove:NTF ! {
3633     \peek_charcode_remove:NTF ! {
3634       \peek_charcode:NTF [ {

```



```

3635     \__stex_variables_invoke_op_custom:nw
3636   }{
3637     % TODO throw error
3638   }
3639 }{
3640   \__stex_variables_invoke_op:n { #1 }
3641 }
3642 }{
3643   \peek_charcode_remove:NTF * {
3644     \__stex_variables_invoke_text:n { #1 }
3645   }{
3646     \__stex_variables_invoke_math_ii:n { #1 }
3647   }
3648 }
3649 }
3650
3651 \cs_new_protected:Nn \__stex_variables_invoke_op:n {
3652   \cs_if_exist:cTF {
3653     stex_var_op_notation_ #1 _cs
3654   }{
3655     \exp_args:Nnx \use:nn {
3656       \def\comp{\_varcomp}
3657       \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3658       \_stex_term_omv:nn { var://#1 }{
3659         \use:c{stex_var_op_notation_ #1 _cs }
3660       }
3661     }{
3662       \_stex_reset:N \comp
3663       \_stex_reset:N \l_stex_current_symbol_str
3664     }
3665   }{
3666     \int_compare:nNnTF {\prop_item:cn {\l_stex_variable_#1_prop}{arity}} = 0{
3667       \__stex_variables_invoke_math_ii:n {#1}
3668     }{
3669       \msg_error:nnxx{stex}{error/noop}{variable~#1}{-}
3670     }
3671   }
3672 }
3673
3674 \cs_new_protected:Npn \__stex_variables_invoke_math_ii:n #1 {
3675   \cs_if_exist:cTF {
3676     stex_var_notation_#1_cs
3677   }{
3678     \tl_set:Nx \stex_symbol_after_invokation_tl {
3679       \_stex_reset:N \comp
3680       \_stex_reset:N \stex_symbol_after_invokation_tl
3681       \_stex_reset:N \l_stex_current_symbol_str
3682       \bool_set_true:N \l_stex_allow_semantic_bool
3683     }
3684     \def\comp{\_varcomp}
3685     \str_set:Nn \l_stex_current_symbol_str { var://#1 }
3686     \bool_set_false:N \l_stex_allow_semantic_bool
3687     \use:c{stex_var_notation_#1_cs}
3688   }{

```

```

3689 \msg_error:nxx{stex}{error/nonotation}{variable~#1}{s}
3690 }
3691 }

```

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

30.5 Sequences

```

3692 <@@=stex_sequences>
3693
3694 \cs_new_protected:Nn \stex_invoke_sequence:n {
3695   \peek_charcode_remove:NTF ! {
3696     \stex_term_omv:nn {varseq://#1}{
3697       \exp_args:Nnx \use:nn {
3698         \def\comp{\_varcomp}
3699         \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3700         \prop_item:cn{stex_varseq_#1_prop}{notation}
3701       }{
3702         \stex_reset:N \comp
3703         \stex_reset:N \l_stex_current_symbol_str
3704       }
3705     }
3706   }{
3707     \bool_set_false:N \l_stex_allow_semantic_bool
3708     \def\comp{\_varcomp}
3709     \str_set:Nn \l_stex_current_symbol_str {varseq://#1}
3710     \tl_set:Nx \stex_symbol_after_invokation_tl {
3711       \stex_reset:N \comp
3712       \stex_reset:N \stex_symbol_after_invokation_tl
3713       \stex_reset:N \l_stex_current_symbol_str
3714       \bool_set_true:N \l_stex_allow_semantic_bool
3715     }
3716     \use:c { stex_varseq_#1_cs }
3717   }
3718 }
3719 </package>

```

Chapter 31

STEX -Structural Features Implementation

```
3720 <*package>
3721
3722 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3723
3724
3725 Warnings and error messages
3726 \msg_new:nnn{stex}{error/copymodule/notallowed}{
3727   Symbol~#1~can~not~be~assigned~in~copymodule~#2
3728 }
3729 \msg_new:nnn{stex}{error/interpretmodule/noddefinens}{
3730   Symbol~#1~not~assigned~in~interpretmodule~#2
3731 }
3732 \msg_new:nnn{stex}{error/unknownstructure}{
3733   No~structure~#1~found!
3734 }
3735 \msg_new:nnn{stex}{error/unknownfield}{
3736   No~field~#1~in~instance~#2~found!~\#3
3737 }
3738 \msg_new:nnn{stex}{error/keyval}{
3739   Invalid~key=value~pair~#1
3740 }
3741 \msg_new:nnn{stex}{error/instantiate/missing}{
3742   Assignments~missing~in~instantiate:~#1
3743 }
3744 \msg_new:nnn{stex}{error/incompatible}{
3745   Incompatible~signature:~#1~(#2)~and~#3~(#4)
3746 }
3747
3748
```

31.1 Imports with modification

```

3749 <@@=stex_copymodule>
3750 \cs_new_protected:Nn \stex_get_symbol_in_seq:nn {
3751   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
3752     \tl_set:Nn \l_tmpa_tl { #1 }
3753     \__stex_copymodule_get_symbol_from_cs:
3754   }{
3755     % argument is a string
3756     % is it a command name?
3757     \cs_if_exist:cTF { #1 }{
3758       \cs_set_eq:Nc \l_tmpa_tl { #1 }
3759       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
3760       \str_if_empty:NNTF \l_tmpa_str {
3761         \exp_args:Nx \cs_if_eq:NNTF {
3762           \tl_head:N \l_tmpa_tl
3763         } \stex_invoke_symbol:n {
3764           \__stex_copymodule_get_symbol_from_cs:n{ #2 }
3765         }{
3766           \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3767         }
3768       } {
3769         \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3770       }
3771     }{
3772       % argument is not a command name
3773       \__stex_copymodule_get_symbol_from_string:nn { #1 }{ #2 }
3774       % \l_stex_all_symbols_seq
3775     }
3776   }
3777 }
3778
3779 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_string:nn {
3780   \str_set:Nn \l_tmpa_str { #1 }
3781   \bool_set_false:N \l_tmpa_bool
3782   \bool_if:NF \l_tmpa_bool {
3783     \tl_set:Nn \l_tmpa_tl {
3784       \msg_error:nnn{stex}{error/unknownsymbol}{#1}
3785     }
3786     \str_set:Nn \l_tmpa_str { #1 }
3787     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
3788     \seq_map_inline:Nn #2 {
3789       \str_set:Nn \l_tmpb_str { ##1 }
3790       \str_if_eq:eeT { \l_tmpa_str } {
3791         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
3792       } {
3793         \seq_map_break:n {
3794           \tl_set:Nn \l_tmpa_tl {
3795             \str_set:Nn \l_stex_get_symbol_uri_str {
3796               ##1
3797             }
3798           }
3799         }
3800       }

```

```

3801     }
3802     \l_tmpa_tl
3803   }
3804 }
3805
3806 \cs_new_protected:Nn \__stex_copymodule_get_symbol_from_cs:n {
3807   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
3808     { \tl_tail:N \l_tmpa_tl }
3809   \tl_if_single:NTF \l_tmpa_tl {
3810     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
3811       \exp_after:wN \str_set:Nn \exp_after:wN
3812         \l_stex_get_symbol_uri_str \l_tmpa_tl
3813       \__stex_copymodule_get_symbol_check:n { #1 }
3814     }{
3815       % TODO
3816       % tail is not a single group
3817     }
3818   }{
3819     % TODO
3820     % tail is not a single group
3821   }
3822 }
3823
3824 \cs_new_protected:Nn \__stex_copymodule_get_symbol_check:n {
3825   \exp_args:NNx \seq_if_in:NnF #1 \l_stex_get_symbol_uri_str {
3826     \msg_error:nxxx{stex}{error/copymodule/notallowed}{\l_stex_get_symbol_uri_str}{
3827       :~\seq_use:Nn #1 {,~}
3828     }
3829   }
3830 }
3831
3832 \cs_new_protected:Nn \stex_copymodule_start:nnnn {
3833   \stex_import_module_uri:nn { #1 } { #2 }
3834   \str_set:Nx \l_stex_current_copymodule_name_str {#3}
3835   \stex_import_require_module:nnnn
3836     { \l_stex_import_ns_str } { \l_stex_import_archive_str }
3837     { \l_stex_import_path_str } { \l_stex_import_name_str }
3838   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
3839   \seq_set_eq:NN \l__stex_copymodule_copymodule_modules_seq \l_stex_collect_imports_seq
3840   \seq_clear:N \l__stex_copymodule_copymodule_fields_seq
3841   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3842     \seq_map_inline:cn {c_stex_module_###_constants}{
3843       \exp_args:NNx \seq_put_right:Nn \l__stex_copymodule_copymodule_fields_seq {
3844         ##1 ? ####1
3845       }
3846     }
3847   }
3848   \seq_clear:N \l_tmpa_seq
3849   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_copymodule_prop {
3850     name      = \l_stex_current_copymodule_name_str ,
3851     module    = \l_stex_current_module_str ,
3852     from      = \l_stex_import_ns_str ?\l_stex_import_name_str ,
3853     includes  = \l_tmpa_seq ,
3854     fields    = \l_tmpa_seq

```

```

3855 }
3856 \stex_debug:nn{copymodule}{#4~for~module~{\l_stex_import_ns_str ?\l_stex_import_name_str}
3857   as~\l_stex_current_module_str?\l_stex_current_copymodule_name_str}
3858   \stex_debug:nn{copymodule}{modules:\seq_use:Nn \l__stex_copymodule_copymodule_modules_seq {,
3859 \stex_debug:nn{copymodule}{fields:\seq_use:Nn \l__stex_copymodule_copymodule_fields_seq {,
3860 \stex_if_smsmode:F {
3861   \begin{stex_annotate_env} {#4} {
3862     \l_stex_current_module_str?\l_stex_current_copymodule_name_str
3863   }
3864   \stex_annotate_invisible:nnn{from}{\l_stex_import_ns_str ?\l_stex_import_name_str}{}
3865 }
3866 \bool_set_eq:NN \l__stex_copymodule_oldhtml_bool \_stex_html_do_output_bool
3867 \bool_set_false:N \_stex_html_do_output_bool
3868 }
3869 \cs_new_protected:Nn \stex_copymodule_end:n {
3870   \def \l_tmpa_cs ##1 ##2 {#1}
3871   \bool_set_eq:NN \_stex_html_do_output_bool \l__stex_copymodule_oldhtml_bool
3872   \tl_clear:N \l_tmpa_tl
3873   \tl_clear:N \l_tmpb_tl
3874   \prop_get:NnN \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3875   \seq_map_inline:Nn \l__stex_copymodule_copymodule_modules_seq {
3876     \seq_map_inline:cn {c_stex_module_##1_constants}{
3877       \tl_clear:N \l_tmpc_tl
3878       \l_tmpa_cs{##1}{####1}
3879       \str_if_exist:cTF {l__stex_copymodule_copymodule_##1?####1_name_str} {
3880         \tl_put_right:Nx \l_tmpa_tl {
3881           \prop_set_from_keyval:cn {
3882             l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3883           }{
3884             \exp_after:wN \prop_to_keyval:N \csname
3885               l_stex_symdecl_\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3886             \endcsname
3887           }
3888           \seq_clear:c {
3889             l_stex_symdecl_
3890             \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3891             _notations
3892           }
3893         }
3894       }
3895       \tl_put_right:Nx \l_tmpc_tl {
3896         \stex_copy_notations:nn {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3897         \stex_annotate_invisible:nnn{alias}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3898       }
3899       \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3900       \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3901         \tl_put_right:Nx \l_tmpc_tl {
3902           \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3903         }
3904         \tl_put_right:Nx \l_tmpa_tl {
3905           \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3906             \stex_invoke_symbol:n {
3907               \l_stex_current_module_str ? \use:c{l__stex_copymodule_copymodule_##1?####1_name_str}
3908             }
3909           }
3910         }
3911       }
3912     }
3913   }

```

```

3909     }
3910   }
3911 }{
3912   \tl_put_right:Nx \l_tmpc_tl {
3913     \stex_copy_notations:nn {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3914   }
3915   \prop_set_eq:Nc \l_tmpa_prop {l_stex_symdecl_ ##1?####1 _prop}
3916   \prop_put:Nnx \l_tmpa_prop { name }{ \l_stex_current_copymodule_name_str / ####1 }
3917   \prop_put:Nnx \l_tmpa_prop { module }{ \l_stex_current_module_str }
3918   \tl_put_right:Nx \l_tmpa_tl {
3919     \prop_set_from_keyval:cn {
3920       l_stex_symdecl_ \l_stex_current_module_str ? \l_stex_current_copymodule_name_str
3921     }{
3922       \prop_to_keyval:N \l_tmpa_prop
3923     }
3924     \seq_clear:c {
3925       l_stex_symdecl_
3926       \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3927       _notations
3928     }
3929   }
3930   \seq_put_right:Nx \l_tmpa_seq {\l_stex_current_module_str ? \l_stex_current_copymodule_name_str}
3931   \str_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_macroname_str} {
3932     \tl_put_right:Nx \l_tmpc_tl {
3933       \stex_annotate_invisible:nnn{macroname}{\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}
3934     }
3935     \tl_put_right:Nx \l_tmpa_tl {
3936       \tl_set:cx {\use:c{l__stex_copymodule_copymodule_##1?####1_macroname_str}}{
3937         \stex_invoke_symbol:n {
3938           \l_stex_current_module_str ? \l_stex_current_copymodule_name_str / ####1
3939         }
3940       }
3941     }
3942   }
3943 }
3944 \tl_if_exist:cT {l__stex_copymodule_copymodule_##1?####1_def_tl}{
3945   \tl_put_right:Nx \l_tmpc_tl {
3946     \stex_annotate_invisible:nnn{definiens}{\use:c{l__stex_copymodule_copymodule_##1?####1_def_tl}}
3947   }
3948 }
3949 \tl_put_right:Nx \l_tmpb_tl {
3950   \stex_annotate:nnn{assignment} {##1?####1} { \l_tmpc_tl }
3951 }
3952 }
3953 }
3954 \prop_put:Nno \l_stex_current_copymodule_prop {fields} \l_tmpa_seq
3955 \tl_put_left:Nx \l_tmpa_tl {
3956   \prop_set_from_keyval:cn {
3957     l_stex_copymodule_ \l_stex_current_module_str? \l_stex_current_copymodule_name_str _prop
3958   }{
3959     \prop_to_keyval:N \l_stex_current_copymodule_prop
3960   }
3961 }
3962 \exp_args:No \stex_add_to_current_module:n \l_tmpa_tl

```

```

3963 \stex_debug:nn{copymodule}{result:\meaning \l_tmpa_tl}
3964 \exp_args:Nx \stex_do_up_to_module:n {
3965   \exp_args:No \exp_not:n \l_tmpa_tl
3966 }
3967 \l_tmpb_tl
3968 \stex_if_smsmode:F {
3969   \end{stex_annotate_env}
3970 }
3971 }
3972
3973 \NewDocumentEnvironment {copymodule} { 0{} m m}{
3974   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ structure }
3975   \stex_deactivate_macro:Nn \symdecl {module~environments}
3976   \stex_deactivate_macro:Nn \symdef {module~environments}
3977   \stex_deactivate_macro:Nn \notation {module~environments}
3978   \stex_reactivate_macro:N \assign
3979   \stex_reactivate_macro:N \renamedekl
3980   \stex_reactivate_macro:N \donotcopy
3981   \stex_smsmode_do:
3982 }{
3983   \stex_copymodule_end:n {}
3984 }
3985
3986 \NewDocumentEnvironment {interpretmodule} { 0{} m m}{
3987   \stex_copymodule_start:nnnn { #1 }{ #2 }{ #3 }{ realization }
3988   \stex_deactivate_macro:Nn \symdecl {module~environments}
3989   \stex_deactivate_macro:Nn \symdef {module~environments}
3990   \stex_deactivate_macro:Nn \notation {module~environments}
3991   \stex_reactivate_macro:N \assign
3992   \stex_reactivate_macro:N \renamedekl
3993   \stex_reactivate_macro:N \donotcopy
3994   \stex_smsmode_do:
3995 }{
3996   \stex_copymodule_end:n {
3997     \tl_if_exist:cF {
3998       l__stex_copymodule_copymodule_##1?##2_def_tl
3999     }{
4000       \str_if_eq:eeF {
4001         \prop_item:cn{
4002           l_stex_symdecl_ ##1 ? ##2 _prop }{ defined }
4003         }{ true }{
4004           \msg_error:nxxx{stex}{error/interpretmodule/nodéfiniens}{
4005             ##1?##2
4006           }{\l_stex_current_copymodule_name_str}
4007         }
4008       }
4009     }
4010   }
4011
4012 \NewDocumentCommand \donotcopy { 0{} m}{
4013   \stex_import_module_uri:nn { #1 } { #2 }
4014   \stex_collect_imports:n {\l_stex_import_ns_str ?\l_stex_import_name_str }
4015   \seq_map_inline:Nn \l_stex_collect_imports_seq {
4016     \seq_remove_all:Nn \l__stex_copymodule_copymodule_modules_seq { ##1 }

```



```

4017 \seq_map_inline:cn {c_stex_module_##1_constants}{
4018 \seq_remove_all:Nn \l__stex_copymodule_copymodule_fields_seq { ##1 ? #####1 }
4019 \bool_lazy_any:nT {
4020 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_name_str}}
4021 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_macroname_str}}
4022 { \cs_if_exist_p:c {l__stex_copymodule_copymodule_##1?####1_def_tl}}
4023 }{
4024 % TODO throw error
4025 }
4026 }
4027 }
4028
4029 \prop_get:NnN \l_stex_current_copymodule_prop { includes } \l_tmpa_seq
4030 \seq_put_right:Nx \l_tmpa_seq {\l_stex_import_ns_str ?\l_stex_import_name_str }
4031 \prop_put:Nno \l_stex_current_copymodule_prop {includes} \l_tmpa_seq
4032 }
4033
4034 \NewDocumentCommand \assign { m m }{
4035 \stex_get_symbol_in_seq:nn {#1} \l__stex_copymodule_copymodule_fields_seq
4036 \stex_debug:nn{assign}{defining~{\l_stex_get_symbol_uri_str}~as~\detokenize{#2}}
4037 \tl_set:cn {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _def_tl}{#2}
4038 }
4039
4040 \keys_define:nn { stex / renamedec1 } {
4041 name .str_set_x:N = \l_stex_renamedec1_name_str
4042 }
4043 \cs_new_protected:Nn \__stex_copymodule_renamedec1_args:n {
4044 \str_clear:N \l_stex_renamedec1_name_str
4045 \keys_set:nn { stex / renamedec1 } { #1 }
4046 }
4047
4048 \NewDocumentCommand \renamedec1 { 0{} m m }{
4049 \__stex_copymodule_renamedec1_args:n { #1 }
4050 \stex_get_symbol_in_seq:nn {#2} \l__stex_copymodule_copymodule_fields_seq
4051 \stex_debug:nn{renamedec1}{renaming~{\l_stex_get_symbol_uri_str}~to~#3}
4052 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _macroname_str}{#3}
4053 \str_if_empty:NTF \l_stex_renamedec1_name_str {
4054 \tl_set:cx { #3 }{\stex_invoke_symbol:n {
4055 \l_stex_get_symbol_uri_str
4056 } }
4057 } {
4058 \str_set:cx {l__stex_copymodule_copymodule_\l_stex_get_symbol_uri_str _name_str}{\l_stex
4059 \stex_debug:nn{renamedec1}{@~\l_stex_current_module_str ? \l_stex_renamedec1_name_str}
4060 \prop_set_eq:cc {l_stex_symdec1_
4061 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4062 _prop
4063 }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _prop}
4064 \seq_set_eq:cc {l_stex_symdec1_
4065 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4066 _notations
4067 }{l_stex_symdec1_ \l_stex_get_symbol_uri_str _notations}
4068 \prop_put:cnx {l_stex_symdec1_
4069 \l_stex_current_module_str ? \l_stex_renamedec1_name_str
4070 _prop

```

```

4071   }{ name }{ \l_stex_renamedekl_name_str }
4072   \prop_put:cnx {l_stex_symdecl_
4073     \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4074     _prop
4075   }{ module }{ \l_stex_current_module_str }
4076   \exp_args:NNx \seq_put_left:Nn \l__stex_copymodule_copymodule_fields_seq {
4077     \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4078   }
4079   \tl_set:cx { #3 }{ \stex_invoke_symbol:n {
4080     \l_stex_current_module_str ? \l_stex_renamedekl_name_str
4081   } }
4082 }
4083 }
4084
4085 \stex_deactivate_macro:Nn \assign {copymodules}
4086 \stex_deactivate_macro:Nn \renamedekl {copymodules}
4087 \stex_deactivate_macro:Nn \donotcopy {copymodules}
4088
4089
4090 \seq_new:N \l_stex_implicit_morphisms_seq
4091 \NewDocumentCommand \implicitmorphism { 0{} m m }{
4092   \stex_import_module_uri:nn { #1 } { #2 }
4093   \stex_debug:nn{implicits}{
4094     Implicit~morphism:~
4095     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4096   }
4097   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
4098     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4099   }{
4100     \msg_error:nnn{stex}{error/conflictingmodules}{
4101       \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4102     }
4103   }
4104
4105   % TODO
4106
4107
4108
4109   \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
4110     \l_stex_module_ns_str ? \l__stex_copymodule_name_str
4111   }
4112 }
4113

```

31.2 The feature environment

structural@feature

```

4114 <@@=stex_features>
4115
4116 \NewDocumentEnvironment{structural_feature_module}{ m m m }{
4117   \stex_if_in_module:F {
4118     \msg_set:nnn{stex}{error/nomodule}{
4119       Structural~Feature~has~to~occur~in~a~module:\\

```

```

4120     Feature~#2~of~type~#1\\
4121     In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
4122   }
4123   \msg_error:nn{stex}{error/nomodule}
4124 }
4125
4126 \stex_module_setup:nn{meta=NONE}{#2 - #1}
4127
4128 \stex_if_smsmode:F {
4129   \begin{stex_annotate_env}{ feature:#1 }{}
4130   \stex_annotate_invisible:nnn{header}{}{ #3 }
4131 }
4132 }{
4133   \str_gset_eq:NN \l_stex_last_feature_str \l_stex_current_module_str
4134   \prop_gput:cnn {c_stex_module_ \l_stex_current_module_str _prop}{feature}{#1}
4135   \stex_debug:nn{features}{
4136     Feature: \l_stex_last_feature_str
4137   }
4138   \stex_if_smsmode:F {
4139     \end{stex_annotate_env}
4140   }
4141 }

```

31.3 Structure

structure

```

4142 <@@=stex_structures>
4143 \cs_new_protected:Nn \stex_add_structure_to_current_module:nn {
4144   \prop_if_exist:cF {c_stex_module_ \l_stex_current_module_str _structures}{
4145     \prop_new:c {c_stex_module_ \l_stex_current_module_str _structures}
4146   }
4147   \prop_gput:cxx{c_stex_module_ \l_stex_current_module_str _structures}
4148   {#1}{#2}
4149 }
4150
4151 \keys_define:nn { stex / features / structure } {
4152   name .str_set_x:N = \l__stex_structures_name_str ,
4153 }
4154
4155 \cs_new_protected:Nn \__stex_structures_structure_args:n {
4156   \str_clear:N \l__stex_structures_name_str
4157   \keys_set:nn { stex / features / structure } { #1 }
4158 }
4159
4160 \NewDocumentEnvironment{mathstructure}{m O{}}{
4161   \__stex_structures_structure_args:n { #2 }
4162   \str_if_empty:NT \l__stex_structures_name_str {
4163     \str_set:Nx \l__stex_structures_name_str { #1 }
4164   }
4165   \exp_args:Nx \stex_symdecl_do:nn {
4166     name = \l__stex_structures_name_str ,
4167     type = \metacollection ,
4168     def = {\STEXsymbol{module-type}}{

```

```

4169         \stex_term_math_oms:nnnn {
4170             \prop_get:cn {c_stex_module_\l_stex_current_module_str _prop}
4171             { ns } ?
4172             \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
4173             { name } / \l__stex_structures_name_str - structure
4174         }{}{0}{}
4175     }}
4176     }{ #1 }
4177     \exp_args:Nnnx
4178     \begin{structural_feature_module}{ structure }
4179     { \l__stex_structures_name_str }{}
4180     \stex_smsmode_do:
4181 }{
4182     \end{structural_feature_module}
4183     \stex_reset_up_to_module:
4184     \exp_args:No \stex_collect_imports:n \l_stex_last_feature_str
4185     \seq_clear:N \l_tmpa_seq
4186     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4187         \seq_map_inline:cn{c_stex_module_##1_constants}{
4188             \seq_put_right:Nn \l_tmpa_seq { ##1 ? ####1 }
4189         }
4190     }
4191     \exp_args:Nnno
4192     \prop_gput:cnn {c_stex_module_ \l_stex_last_feature_str _prop}{fields}\l_tmpa_seq
4193     \stex_debug:nn{structure}{Fields:~\seq_use:Nn \l_tmpa_seq ,}
4194     \stex_add_structure_to_current_module:nn
4195         \l__stex_structures_name_str
4196         \l_stex_last_feature_str
4197     \exp_args:Nx
4198     \stex_add_to_current_module:n {
4199         \tl_set:cn { #1 }{
4200             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4201         }
4202     }
4203     \exp_args:Nx
4204     \stex_do_up_to_module:n {
4205         \tl_set:cn { #1 }{
4206             \exp_not:N \stex_invoke_structure:nn {\l_stex_current_module_str }{ \l__stex_structures
4207         }
4208     }
4209 }
4210 \seq_put_right:Nx \g_stex_smsmode_allowedenvs_seq { \tl_to_str:n {mathstructure}}
4211
4212 \cs_new:Nn \stex_invoke_structure:nn {
4213     \stex_invoke_symbol:n { #1?#2 }
4214 }
4215
4216 \cs_new_protected:Nn \stex_get_structure:n {
4217     \tl_if_head_eq_catcode:nNTF { #1 } \relax {
4218         \tl_set:Nn \l_tmpa_tl { #1 }
4219         \__stex_structures_get_from_cs:
4220     }{
4221         \cs_if_exist:cTF { #1 }{
4222             \cs_set_eq:Nc \l_tmpa_cs { #1 }

```

```

4223 \str_set:Nx \l_tmpa_str {\cs_argument_spec:N \l_tmpa_cs }
4224 \str_if_empty:NTF \l_tmpa_str {
4225   \cs_if_eq:NNTF { \tl_head:N \l_tmpa_cs} \stex_invoke_structure:nn {
4226     \__stex_structures_get_from_cs:
4227   }{
4228     \__stex_structures_get_from_string:n { #1 }
4229   }
4230 }{
4231   \__stex_structures_get_from_string:n { #1 }
4232 }
4233 }{
4234   \__stex_structures_get_from_string:n { #1 }
4235 }
4236 }
4237 }
4238
4239 \cs_new_protected:Nn \__stex_structures_get_from_cs: {
4240   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
4241     { \tl_tail:N \l_tmpa_tl }
4242   \str_set:Nx \l_tmpa_str {
4243     \exp_after:wN \use_i:nn \l_tmpa_tl
4244   }
4245   \str_set:Nx \l_tmpb_str {
4246     \exp_after:wN \use_ii:nn \l_tmpa_tl
4247   }
4248   \str_set:Nx \l_stex_get_structure_str {
4249     \l_tmpa_str ? \l_tmpb_str
4250   }
4251   \str_set:Nx \l_stex_get_structure_module_str {
4252     \exp_args:Nno \prop_item:cn {c_stex_module_\l_tmpa_str _structures}{\l_tmpb_str}
4253   }
4254 }
4255
4256 \cs_new_protected:Nn \__stex_structures_get_from_string:n {
4257   \tl_set:Nn \l_tmpa_tl {
4258     \msg_error:nnn{stex}{error/unknownstructure}{#1}
4259   }
4260   \str_set:Nn \l_tmpa_str { #1 }
4261   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
4262
4263   \seq_map_inline:Nn \l_stex_all_modules_seq {
4264     \prop_if_exist:cT {c_stex_module_##1_structures} {
4265       \prop_map_inline:cn {c_stex_module_##1_structures} {
4266         \str_if_eq:eeT { \l_tmpa_str }{ \str_range:nnn {##1?####1}{-\l_tmpa_int}{-1}}{
4267           \prop_map_break:n{\seq_map_break:n{
4268             \tl_set:Nn \l_tmpa_tl {
4269               \str_set:Nn \l_stex_get_structure_str {##1?####1}
4270               \str_set:Nn \l_stex_get_structure_module_str {####2}
4271             }
4272           }}
4273         }
4274       }
4275     }
4276   }

```

```

4277 \l_tmpa_tl
4278 }

\instantiate

4279
4280 \keys_define:nn { stex / instantiate } {
4281   name          .str_set_x:N = \l__stex_structures_name_str
4282 }
4283 \cs_new_protected:Nn \__stex_structures_instantiate_args:n {
4284   \str_clear:N \l__stex_structures_name_str
4285   \keys_set:nn { stex / instantiate } { #1 }
4286 }
4287
4288 \NewDocumentCommand \instantiate {m O{} m m m}{
4289   \beginingroup
4290     \stex_get_structure:n {#4}
4291     \__stex_structures_instantiate_args:n { #2 }
4292     \str_if_empty:NT \l__stex_structures_name_str {
4293       \str_set:Nn \l__stex_structures_name_str { #1 }
4294     }
4295     \seq_clear:N \l__stex_structures_fields_seq
4296     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4297     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4298       \seq_map_inline:cn {c_stex_module_##1_constants}{
4299         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4300       }
4301     }
4302     \seq_set_split:Nnn \l_tmpa_seq , {#3}
4303     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4304     \prop_clear:N \l_tmpa_prop
4305     \seq_map_inline:Nn \l_tmpa_seq {
4306       \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4307       \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4308         \msg_error:nnn{stex}{error/keyval}{##1}
4309       }
4310       \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_structur
4311       \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4312       \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_uri
4313       \exp_args:Nx \stex_get_symbol:n {\seq_item:Nn \l_tmpb_seq 2}
4314       \exp_args:Nxx \str_if_eq:nnF
4315         {\prop_item:cn{l_stex_symdecl \l__stex_structures_dom_str _prop}{args}}
4316         {\prop_item:cn{l_stex_symdecl \l_stex_get_symbol_uri_str _prop}{args}}{
4317         \msg_error:nnxxx{stex}{error/incompatible}
4318         {\l__stex_structures_dom_str
4319         {\prop_item:cn{l_stex_symdecl \l__stex_structures_dom_str _prop}{args}}
4320         {\l_stex_get_symbol_uri_str
4321         {\prop_item:cn{l_stex_symdecl \l_stex_get_symbol_uri_str _prop}{args}}
4322       }
4323       \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} \l_stex_get_symbol_uri_str
4324     }
4325     \seq_if_empty:NF \l__stex_structures_fields_seq {
4326       \msg_error:nnx{stex}{error/instantiate/missing}{\seq_use:Nn \l__stex_structures_fields_
4327     }
4328     \exp_args:Nx

```

```

4329 \stex_add_to_current_module:n {
4330   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4331     domain = \l_stex_get_structure_module_str ,
4332     \prop_to_keyval:N \l_tmpa_prop
4333   }
4334   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4335 }
4336 \exp_args:Nx
4337 \stex_do_up_to_module:n {
4338   \prop_set_from_keyval:cn {l_stex_instance_\l_stex_current_module_str?\l__stex_structur
4339     domain = \l_stex_get_structure_module_str ,
4340     \prop_to_keyval:N \l_tmpa_prop
4341   }
4342   \tl_set:cn{ #1 }{\stex_invoke_instance:n{ \l_stex_current_module_str?\l__stex_structur
4343 }
4344 \stex_debug:nn{instantiate}{
4345   Instance~\l_stex_current_module_str?\l__stex_structures_name_str \
4346   \prop_to_keyval:N \l_tmpa_prop
4347 }
4348 \exp_args:Nxx \stex_symdecl_do:nn {
4349   type={\STEXsymbol{module-type}}{
4350     \stex_term_math_oms:nnnn {
4351       \l_stex_get_structure_module_str
4352     }{}{0}{}
4353   }}
4354   {}{\l__stex_structures_name_str}
4355   \exp_args:Nx \notation{\l__stex_structures_name_str}{\comp{#5}}
4356 \endgroup
4357 \stex_smsmode_do:\ignorespacesandpars
4358 }
4359 \tl_put_right:Nx \g_stex_smsmode_allowedmacros_escape_tl {\instantiate}
4360
4361 \cs_new_protected:Nn \stex_symbol_or_var:n {
4362   \cs_if_exist:cTF{#1}{
4363     \cs_set_eq:Nc \l_tmpa_tl { #1 }
4364     \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
4365     \str_if_empty:NTF \l_tmpa_str {
4366       \exp_args:Nx \cs_if_eq:NNTF { \tl_head:N \l_tmpa_tl }
4367         \stex_invoke_variable:n {
4368           \bool_set_true:N \l_stex_symbol_or_var_bool
4369           \tl_set:Nx \l_tmpa_tl {\tl_tail:N \l_tmpa_tl}
4370           \str_set:Nx \l_stex_get_symbol_uri_str {
4371             \exp_after:wN \use:n \l_tmpa_tl
4372           }
4373         }{
4374           \bool_set_false:N \l_stex_symbol_or_var_bool
4375           \stex_get_symbol:n{#1}
4376         }
4377       }{
4378         \__stex_structures_symbolorvar_from_string:n{ #1 }
4379       }
4380     }{
4381       \__stex_structures_symbolorvar_from_string:n{ #1 }
4382     }

```

```

4383 }
4384
4385 \cs_new_protected:Nn \__stex_structures_symbolorvar_from_string:n {
4386   \prop_if_exist:cTF {l_stex_variable_#1 _prop}{
4387     \bool_set_true:N \l_stex_symbol_or_var_bool
4388     \str_set:Nn \l_stex_get_symbol_uri_str { #1 }
4389   }{
4390     \bool_set_false:N \l_stex_symbol_or_var_bool
4391     \stex_get_symbol:n{#1}
4392   }
4393 }
4394
4395 \keys_define:nn { stex / varinstantiate } {
4396   name .str_set_x:N = \l__stex_structures_name_str,
4397   bind .choices:nn =
4398     {forall,exists}
4399     {\str_set:Nx \l__stex_structures_bind_str {\l_keys_choice_tl}}
4400 }
4401
4402 \cs_new_protected:Nn \__stex_structures_varinstantiate_args:n {
4403   \str_clear:N \l__stex_structures_name_str
4404   \str_clear:N \l__stex_structures_bind_str
4405   \keys_set:nn { stex / varinstantiate } { #1 }
4406 }
4407
4408 \NewDocumentCommand \varinstantiate {m O{}} m m m m){
4409   \beginingroup
4410     \stex_get_structure:n {#4}
4411     \__stex_structures_varinstantiate_args:n { #2 }
4412     \str_if_empty:NT \l__stex_structures_name_str {
4413       \str_set:Nn \l__stex_structures_name_str { #1 }
4414     }
4415     \seq_clear:N \l__stex_structures_fields_seq
4416     \exp_args:Nx \stex_collect_imports:n \l_stex_get_structure_module_str
4417     \seq_map_inline:Nn \l_stex_collect_imports_seq {
4418       \seq_map_inline:cn {c_stex_module_##1_constants}{
4419         \seq_put_right:Nx \l__stex_structures_fields_seq { ##1 ? #####1 }
4420       }
4421     }
4422     \exp_args:No \stex_activate_module:n \l_stex_get_structure_module_str
4423     \prop_clear:N \l_tmpa_prop
4424     \tl_if_empty:nF {#3} {
4425       \seq_set_split:Nnn \l_tmpa_seq , {#3}
4426       \seq_map_inline:Nn \l_tmpa_seq {
4427         \seq_set_split:Nnn \l_tmpb_seq = { ##1 }
4428         \int_compare:nNnF { \seq_count:N \l_tmpb_seq } = 2 {
4429           \msg_error:nnn{stex}{error/keyval}{##1}
4430         }
4431         \exp_args:Nx \stex_get_symbol_in_seq:nn {\seq_item:Nn \l_tmpb_seq 1} \l__stex_struct
4432         \str_set_eq:NN \l__stex_structures_dom_str \l_stex_get_symbol_uri_str
4433         \exp_args:NNx \seq_remove_all:Nn \l__stex_structures_fields_seq \l_stex_get_symbol_u
4434         \exp_args:Nx \stex_symbol_or_var:n {\seq_item:Nn \l_tmpb_seq 2}
4435         \bool_if:NTF \l_stex_symbol_or_var_bool {
4436           \exp_args:Nxx \str_if_eq:nnF

```



```

4437         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4438         {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}{
4439         \msg_error:nnxxxx{stex}{error/incompatible}
4440         {\l__stex_structures_dom_str}
4441         {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4442         {\l_stex_get_symbol_uri_str}
4443         {\prop_item:cn{l_stex_variable_\l_stex_get_symbol_uri_str _prop}{args}}
4444     }
4445     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_variable:n {
4446 }}{
4447     \exp_args:Nxx \str_if_eq:nnF
4448     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4449     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}{
4450     \msg_error:nnxxxx{stex}{error/incompatible}
4451     {\l__stex_structures_dom_str}
4452     {\prop_item:cn{l_stex_symdecl_\l__stex_structures_dom_str _prop}{args}}
4453     {\l_stex_get_symbol_uri_str}
4454     {\prop_item:cn{l_stex_symdecl_\l_stex_get_symbol_uri_str _prop}{args}}
4455     }
4456     \prop_put:Nxx \l_tmpa_prop {\seq_item:Nn \l_tmpb_seq 1} {\stex_invoke_symbol:n {\l
4457 }}
4458 }
4459 }
4460 \tl_gclear:N \g__stex_structures_aftergroup_tl
4461 \seq_map_inline:Nn \l__stex_structures_fields_seq {
4462     \str_set:Nx \l_tmpa_str {\l__stex_structures_name_str . \prop_item:cn {l_stex_symdecl_\l
4463     \stex_debug:nn{varinstantiate}{Field~\l_tmpa_str :~##1}
4464     \seq_if_empty:cF{l_stex_symdecl_##1_notations}{
4465         \stex_find_notation:nn{##1}{}
4466         \cs_gset_eq:cc{g__stex_structures_tmpa_\l_tmpa_str _cs}
4467         {stex_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4468         \stex_debug:nn{varinstantiate}{Notation:~\cs_meaning:c{g__stex_structures_tmpa_\l_tm
4469         \cs_if_exist:cT{stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}{
4470         \cs_gset_eq:cc {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4471         {stex_op_notation_##1\c_hash_str \l_stex_notation_variant_str _cs}
4472         \stex_debug:nn{varinstantiate}{Operator~Notation:~\cs_meaning:c{g__stex_structur
4473     }
4474 }
4475 }
4476 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {
4477     \prop_set_from_keyval:cn { l_stex_variable_ \l_tmpa_str _prop}{
4478         name = \l_tmpa_str ,
4479         args = \prop_item:cn {l_stex_symdecl_##1_prop}{args} ,
4480         arity = \prop_item:cn {l_stex_symdecl_##1_prop}{arity} ,
4481         assocs = \prop_item:cn {l_stex_symdecl_##1_prop}{assocs}
4482     }
4483     \cs_set_eq:cc {stex_var_notation_\l_tmpa_str _cs}
4484     {g__stex_structures_tmpa_\l_tmpa_str _cs}
4485     \cs_set_eq:cc {stex_var_op_notation_\l_tmpa_str _cs}
4486     {g__stex_structures_tmpa_op_\l_tmpa_str _cs}
4487 }
4488 \prop_put:Nxx \l_tmpa_prop {\prop_item:cn {l_stex_symdecl_##1_prop}{name}}{\stex_invok
4489 }
4490 \exp_args:NNx \tl_gput_right:Nn \g__stex_structures_aftergroup_tl {

```

```

4491 \prop_set_from_keyval:cn {l_stex_varinstance\_l__stex_structures_name_str _prop }{
4492   domain = \l_stex_get_structure_module_str ,
4493   \prop_to_keyval:N \l_tmpa_prop
4494 }
4495 \tl_set:cn { #1 }{\stex_invoke_varinstance:n {\l__stex_structures_name_str}}
4496 \tl_set:cn {l_stex_varinstance\_l__stex_structures_name_str _op_tl}{
4497   \exp_args:Nnx \exp_not:N \use:nn {
4498     \str_set:Nn \exp_not:N \l_stex_current_symbol_str {var://\l__stex_structures_name_
4499     \_stex_term_omv:nn {var://\l__stex_structures_name_str}{
4500       \exp_not:n{
4501         \_varcomp{#5}
4502       }
4503     }
4504   }{
4505     \exp_not:n{\_stex_reset:N \l_stex_current_symbol_str}
4506   }
4507 }
4508 }
4509 \stex_debug:nn{varianstantiate}{\expandafter\detokenize\expandafter{\g__stex_structures_
4510 \aftergroup\g__stex_structures_aftergroup_tl
4511 \endgroup
4512 \stex_smsmode_do:\ignorespacesandpars
4513 }
4514
4515 \cs_new_protected:Nn \stex_invoke_instance:n {
4516   \peek_charcode_remove:NTF ! {
4517     \stex_invoke_symbol:n{#1}
4518   }{
4519     \_stex_invoke_instance:nn {#1}
4520   }
4521 }
4522
4523
4524 \cs_new_protected:Nn \stex_invoke_varinstance:n {
4525   \peek_charcode_remove:NTF ! {
4526     \exp_args:Nnx \use:nn {
4527       \def\comp{\_varcomp}
4528       \use:c{l_stex_varinstance_#1_op_tl}
4529     }{
4530       \_stex_reset:N \comp
4531     }
4532   }{
4533     \_stex_invoke_varinstance:nn {#1}
4534   }
4535 }
4536
4537 \cs_new_protected:Nn \_stex_invoke_instance:nn {
4538   \prop_if_in:cnTF {l_stex_instance_ #1 _prop}{#2}{
4539     \exp_args:Nx \stex_invoke_symbol:n {\prop_item:cn{l_stex_instance_ #1 _prop}{#2}}
4540   }{
4541     \prop_set_eq:Nc \l_tmpa_prop{l_stex_instance_ #1 _prop}
4542     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{
4543       \prop_to_keyval:N \l_tmpa_prop
4544     }

```

```

4545 }
4546 }
4547
4548 \cs_new_protected:Nn \stex_invoke_varinstance:nn {
4549   \prop_if_in:cnTF {l_stex_varinstance_ #1 _prop}{#2}{
4550     \prop_get:cnN{l_stex_varinstance_ #1 _prop}{#2}\l_tmpa_tl
4551     \l_tmpa_tl
4552   }{
4553     \msg_error:nnnn{stex}{error/unknownfield}{#2}{#1}{ }
4554   }
4555 }

```

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

\stex_invoke_structure:nnn

```

4556 % #1: URI of the instance
4557 % #2: URI of the instantiated module
4558 \cs_new_protected:Nn \stex_invoke_structure:nnn {
4559   \tl_if_empty:nTF{ #3 }{
4560     \prop_set_eq:Nc \l__stex_structures_structure_prop {
4561       c_stex_feature_ #2 _prop
4562     }
4563     \tl_clear:N \l_tmpa_tl
4564     \prop_get:NnN \l__stex_structures_structure_prop { fields } \l_tmpa_seq
4565     \seq_map_inline:Nn \l_tmpa_seq {
4566       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
4567       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
4568       \cs_if_exist:cT {
4569         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
4570       }{
4571         \tl_if_empty:NF \l_tmpa_tl {
4572           \tl_put_right:Nn \l_tmpa_tl {,}
4573         }
4574         \tl_put_right:Nx \l_tmpa_tl {
4575           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
4576         }
4577       }
4578     }
4579     \exp_args:No \mathstruct \l_tmpa_tl
4580   }{
4581     \stex_invoke_symbol:n{#1/#3}
4582   }
4583 }

```

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

```

4584 </package>

```

Chapter 32

STEX -Statements Implementation

```
4585 <*package>
4586
4587 %%%%%%%%%%% features.dtx %%%%%%%%%%%
4588
4589 <@@=stex_statements>
    Warnings and error messages
4590
\titleemph
4591 \def\titleemph#1{\textbf{#1}}
(End definition for \titleemph. This function is documented on page ??.)
```

32.1 Definitions

definiendum

```
4592 \keys_define:nn {stex / definiendum }{
4593   pre      .tl_set:N      = \l__stex_statements_definiendum_pre_tl,
4594   post     .tl_set:N      = \l__stex_statements_definiendum_post_tl,
4595   root     .str_set_x:N    = \l__stex_statements_definiendum_root_str,
4596   gfa      .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
4597 }
4598 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
4599   \str_clear:N \l__stex_statements_definiendum_root_str
4600   \tl_clear:N \l__stex_statements_definiendum_post_tl
4601   \str_clear:N \l__stex_statements_definiendum_gfa_str
4602   \keys_set:nn { stex / definiendum }{ #1 }
4603 }
4604 \NewDocumentCommand \definiendum { O{} m m } {
4605   \__stex_statements_definiendum_args:n { #1 }
4606   \stex_get_symbol:n { #2 }
4607   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4608   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
4609     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
```

```

4610     \tl_set:Nn \l_tmpa_tl { #3 }
4611   } {
4612     \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
4613     \tl_set:Nn \l_tmpa_tl {
4614       \l__stex_statements_definiendum_pre_tl\l__stex_statements_definiendum_root_str\l__st
4615     }
4616   }
4617 } {
4618   \tl_set:Nn \l_tmpa_tl { #3 }
4619 }
4620
4621 % TODO root
4622 \rustex_if:TF {
4623   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
4624 } {
4625   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
4626 }
4627 }
4628 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

4629
4630 \NewDocumentCommand \definame { 0{ } m } {
4631   \__stex_statements_definiendum_args:n { #1 }
4632   % TODO: root
4633   \stex_get_symbol:n { #2 }
4634   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4635   \str_set:Nx \l_tmpa_str {
4636     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4637   }
4638   \str_replace_all:Nnn \l_tmpa_str {-} {~}
4639   \rustex_if:TF {
4640     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4641       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4642     }
4643   } {
4644     \exp_args:Nnx \defemph@uri {
4645       \l_tmpa_str\l__stex_statements_definiendum_post_tl
4646     } { \l_stex_get_symbol_uri_str }
4647   }
4648 }
4649 \stex_deactivate_macro:Nn \definame {definition~environments}
4650
4651 \NewDocumentCommand \Definame { 0{ } m } {
4652   \__stex_statements_definiendum_args:n { #1 }
4653   \stex_get_symbol:n { #2 }
4654   \str_set:Nx \l_tmpa_str {
4655     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
4656   }
4657   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
4658   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
4659   \rustex_if:TF {

```

```

4660 \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
4661 \l_tmpa_str\l__stex_statements_definiendum_post_tl
4662 }
4663 } {
4664 \exp_args:Nnx \defemph@uri {
4665 \exp_after:wN \stex_capitalize:n \l_tmpa_str\l__stex_statements_definiendum_post_tl
4666 } { \l_stex_get_symbol_uri_str }
4667 }
4668 }
4669 \stex_deactivate_macro:Nn \Definame {definition~environments}
4670
4671 \NewDocumentCommand \premise { m }{
4672 \stex_annotate:nnn{ premise }{}{ #1 }
4673 }
4674 \NewDocumentCommand \conclusion { m }{
4675 \stex_annotate:nnn{ conclusion }{}{ #1 }
4676 }
4677 \NewDocumentCommand \definiens { 0{} m }{
4678 \str_clear:N \l_stex_get_symbol_uri_str
4679 \tl_if_empty:nF {#1} {
4680 \stex_get_symbol:n { #1 }
4681 }
4682 \stex_annotate:nnn{ definiens }{\l_stex_get_symbol_uri_str}{ #2 }
4683 }
4684
4685 \stex_deactivate_macro:Nn \premise {definition~example~or~assertion~environments}
4686 \stex_deactivate_macro:Nn \conclusion {example~or~assertion~environments}
4687 \stex_deactivate_macro:Nn \definiens {definition~environments}
4688

```

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

sdefinition

```

4689
4690 \keys_define:nn {stex / sdefinition }{
4691 type .str_set_x:N = \sdefinitiontype,
4692 id .str_set_x:N = \sdefinitionid,
4693 name .str_set_x:N = \sdefinitionname,
4694 for .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4695 title .tl_set:N = \sdefinitiontitle
4696 }
4697 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
4698 \str_clear:N \sdefinitiontype
4699 \str_clear:N \sdefinitionid
4700 \str_clear:N \sdefinitionname
4701 \clist_clear:N \l__stex_statements_sdefinition_for_clist
4702 \tl_clear:N \sdefinitiontitle
4703 \keys_set:nn { stex / sdefinition }{ #1 }
4704 }
4705
4706 \NewDocumentEnvironment{sdefinition}{0{} }{
4707 \__stex_statements_sdefinition_args:n{ #1 }
4708 \stex_reactivate_macro:N \definiendum
4709 \stex_reactivate_macro:N \definame

```

```

4710 \stex_reactivate_macro:N \Definame
4711 \stex_reactivate_macro:N \premise
4712 \stex_reactivate_macro:N \definiens
4713 \stex_if_smsmode:F{
4714   \seq_clear:N \l_tmpa_seq
4715   \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4716     \tl_if_empty:nF{ ##1 }{
4717       \stex_get_symbol:n { ##1 }
4718       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4719         \l_stex_get_symbol_uri_str
4720       }
4721     }
4722   }
4723   \exp_args:Nnnx
4724   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpa_seq {,}}
4725   \str_if_empty:NF \sdefinitiontype {
4726     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4727   }
4728   \clist_set:No \l_tmpa_clist \sdefinitiontype
4729   \tl_clear:N \l_tmpa_tl
4730   \clist_map_inline:Nn \l_tmpa_clist {
4731     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
4732       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
4733     }
4734   }
4735   \tl_if_empty:NTF \l_tmpa_tl {
4736     \__stex_statements_sdefinition_start:
4737   }{
4738     \l_tmpa_tl
4739   }
4740 }
4741 \stex_ref_new_doc_target:n \sdefinitionid
4742 \stex_smsmode_do:
4743 ){
4744   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4745   \stex_if_smsmode:F {
4746     \clist_set:No \l_tmpa_clist \sdefinitiontype
4747     \tl_clear:N \l_tmpa_tl
4748     \clist_map_inline:Nn \l_tmpa_clist {
4749       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
4750         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
4751       }
4752     }
4753     \tl_if_empty:NTF \l_tmpa_tl {
4754       \__stex_statements_sdefinition_end:
4755     }{
4756       \l_tmpa_tl
4757     }
4758     \end{stex_annotate_env}
4759   }
4760 }

```

\stexpatchdefinition

```

4761 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {

```

```

4762 \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
4763   ~(\sdefinitiontitle)
4764 }~}
4765 }
4766 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
4767
4768 \newcommand\stexpatchdefinition[3][] {
4769   \str_set:Nx \l_tmpa_str{ #1 }
4770   \str_if_empty:NTF \l_tmpa_str {
4771     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
4772     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
4773   }{
4774     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
4775     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
4776   }
4777 }

```

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

\inlinedef inline:

```

4778 \keys_define:nn {stex / inlinedef }{
4779   type      .str_set_x:N = \sdefinitiontype,
4780   id        .str_set_x:N = \sdefinitionid,
4781   for       .clist_set:N = \l__stex_statements_sdefinition_for_clist ,
4782   name      .str_set_x:N = \sdefinitionname
4783 }
4784 \cs_new_protected:Nn \__stex_statements_inlinedef_args:n {
4785   \str_clear:N \sdefinitiontype
4786   \str_clear:N \sdefinitionid
4787   \str_clear:N \sdefinitionname
4788   \clist_clear:N \l__stex_statements_sdefinition_for_clist
4789   \keys_set:nn { stex / inlinedef }{ #1 }
4790 }
4791 \NewDocumentCommand \inlinedef { 0{} m } {
4792   \begingroup
4793   \__stex_statements_inlinedef_args:n{ #1 }
4794   \stex_reactivate_macro:N \definiendum
4795   \stex_reactivate_macro:N \definame
4796   \stex_reactivate_macro:N \Definame
4797   \stex_reactivate_macro:N \premise
4798   \stex_reactivate_macro:N \definiens
4799   \stex_ref_new_doc_target:n \sdefinitionid
4800   \stex_if_smsmode:TF{
4801     \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}{\sdefinitionname} }
4802   }{
4803     \seq_clear:N \l_tmpa_seq
4804     \clist_map_inline:Nn \l__stex_statements_sdefinition_for_clist {
4805       \tl_if_empty:nF{ ##1 }{
4806         \stex_get_symbol:n { ##1 }
4807         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4808           \l_stex_get_symbol_uri_str
4809         }
4810       }
4811     }

```



```

4812 \exp_args:Nnx
4813 \stex_annotate:nnn{definition}{\seq_use:Nn \l_tmpa_seq {,}}{
4814   \str_if_empty:NF \sdefinitiontype {
4815     \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
4816   }
4817   #2
4818   \str_if_empty:NF \sdefinitionname { \stex_symdecl_do:nn{}}{\sdefinitionname} }
4819 }
4820 }
4821 \endgroup
4822 \stex_smsmode_do:
4823 }

```

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

32.2 Assertions

sassertion

```

4824
4825 \keys_define:nn {stex / sassertion }{
4826   type .str_set_x:N = \sassertiontype,
4827   id .str_set_x:N = \sassertionid,
4828   title .tl_set:N = \sassertiontitle ,
4829   for .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4830   name .str_set_x:N = \sassertionname
4831 }
4832 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
4833   \str_clear:N \sassertiontype
4834   \str_clear:N \sassertionid
4835   \str_clear:N \sassertionname
4836   \clist_clear:N \l__stex_statements_sassertion_for_clist
4837   \tl_clear:N \sassertiontitle
4838   \keys_set:nn { stex / sassertion }{ #1 }
4839 }
4840
4841 %\tl_new:N \g__stex_statements_aftergroup_tl
4842
4843 \NewDocumentEnvironment{sassertion}{0{}}{
4844   \__stex_statements_sassertion_args:n{ #1 }
4845   \stex_reactivate_macro:N \premise
4846   \stex_reactivate_macro:N \conclusion
4847   \stex_if_smsmode:F {
4848     \seq_clear:N \l_tmpa_seq
4849     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4850       \tl_if_empty:nF{ ##1 }{
4851         \stex_get_symbol:n { ##1 }
4852         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4853           \l_stex_get_symbol_uri_str
4854         }
4855       }
4856     }
4857   }
4858   \exp_args:Nnnx
4859   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpa_seq {,}}

```

```

4859 \str_if_empty:NF \sassertiontype {
4860   \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
4861 }
4862 \clist_set:No \l_tmpa_clist \sassertiontype
4863 \tl_clear:N \l_tmpa_tl
4864 \clist_map_inline:Nn \l_tmpa_clist {
4865   \tl_if_exist:cT {__stex_statements_sassertion_##1_start:}{
4866     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_start:}}
4867   }
4868 }
4869 \tl_if_empty:NTF \l_tmpa_tl {
4870   \__stex_statements_sassertion_start:
4871 }{
4872   \l_tmpa_tl
4873 }
4874 }
4875 \str_if_empty:NTF \sassertionid {
4876   \str_if_empty:NF \sassertionname {
4877     \stex_ref_new_doc_target:n {}
4878   }
4879 } {
4880   \stex_ref_new_doc_target:n \sassertionid
4881 }
4882 \stex_smsmode_do:
4883 }{
4884   \str_if_empty:NF \sassertionname {
4885     \stex_symdecl_do:nn{}{\sassertionname}
4886     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4887   }
4888   \stex_if_smsmode:F {
4889     \clist_set:No \l_tmpa_clist \sassertiontype
4890     \tl_clear:N \l_tmpa_tl
4891     \clist_map_inline:Nn \l_tmpa_clist {
4892       \tl_if_exist:cT {__stex_statements_sassertion_##1_end:}{
4893         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sassertion_##1_end:}}
4894       }
4895     }
4896     \tl_if_empty:NTF \l_tmpa_tl {
4897       \__stex_statements_sassertion_end:
4898     }{
4899       \l_tmpa_tl
4900     }
4901     \end{stex_annotate_env}
4902   }
4903 }

```

\stexpatchassertion

```

4904
4905 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
4906   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {
4907     (\sassertiontitle)
4908   }~}
4909 }
4910 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}

```

```

4911
4912 \newcommand\stexpatchassertion[3] [] {
4913   \str_set:Nx \l_tmpa_str{ #1 }
4914   \str_if_empty:NTF \l_tmpa_str {
4915     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
4916     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
4917   }{
4918     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
4919     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
4920   }
4921 }

```

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

\inlineass inline:

```

4922 \keys_define:nn {stex / inlineass }{
4923   type      .str_set_x:N = \sassertiontype,
4924   id        .str_set_x:N = \sassertionid,
4925   for       .clist_set:N = \l__stex_statements_sassertion_for_clist ,
4926   name      .str_set_x:N = \sassertionname
4927 }
4928 \cs_new_protected:Nn \__stex_statements_inlineass_args:n {
4929   \str_clear:N \sassertiontype
4930   \str_clear:N \sassertionid
4931   \str_clear:N \sassertionname
4932   \clist_clear:N \l__stex_statements_sassertion_for_clist
4933   \keys_set:nn { stex / inlineass }{ #1 }
4934 }
4935 \NewDocumentCommand \inlineass { 0{} m } {
4936   \begingroup
4937   \stex_reactivate_macro:N \premise
4938   \stex_reactivate_macro:N \conclusion
4939   \__stex_statements_inlineass_args:n{ #1 }
4940   \str_if_empty:NTF \sassertionid {
4941     \str_if_empty:NF \sassertionname {
4942       \stex_ref_new_doc_target:n {}
4943     }
4944   } {
4945     \stex_ref_new_doc_target:n \sassertionid
4946   }
4947
4948   \stex_if_smsmode:TF{
4949     \str_if_empty:NF \sassertionname {
4950       \stex_symdecl_do:nn{}{\sassertionname}
4951       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4952     }
4953   }{
4954     \seq_clear:N \l_tmpa_seq
4955     \clist_map_inline:Nn \l__stex_statements_sassertion_for_clist {
4956       \tl_if_empty:nF{ ##1 }{
4957         \stex_get_symbol:n { ##1 }
4958         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
4959           \l_stex_get_symbol_uri_str
4960         }

```

```

4961     }
4962   }
4963   \exp_args:Nnx
4964   \stex_annotate:nnn{assertion}{\seq_use:Nn \l_tmpa_seq {,}}{
4965     \str_if_empty:NF \sassertiontype {
4966       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}}
4967   }
4968   #2
4969   \str_if_empty:NF \sassertionname {
4970     \stex_symdecl_do:nn{}{\sassertionname}
4971     \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sassertionname}
4972   }
4973 }
4974 }
4975 \endgroup
4976 \stex_smsmode_do:
4977 }

```

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

32.3 Examples

`sexample`

```

4978
4979 \keys_define:nn {stex / sexample }{
4980   type      .str_set_x:N = \exampletype,
4981   id        .str_set_x:N = \sexampleid,
4982   title     .tl_set:N     = \sexampletitle,
4983   name      .str_set_x:N = \sexamplename ,
4984   for       .clist_set:N  = \l__stex_statements_sexample_for_clist,
4985 }
4986 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
4987   \str_clear:N \sexampletype
4988   \str_clear:N \sexampleid
4989   \str_clear:N \sexamplename
4990   \tl_clear:N \sexampletitle
4991   \clist_clear:N \l__stex_statements_sexample_for_clist
4992   \keys_set:nn { stex / sexample }{ #1 }
4993 }
4994
4995 \NewDocumentEnvironment{sexample}{0{}}{
4996   \__stex_statements_sexample_args:n{ #1 }
4997   \stex_reactivate_macro:N \premise
4998   \stex_reactivate_macro:N \conclusion
4999   \stex_if_smsmode:F {
5000     \seq_clear:N \l_tmpa_seq
5001     \clist_map_inline:Nn \l__stex_statements_sexample_for_clist {
5002       \tl_if_empty:nF{ ##1 }{
5003         \stex_get_symbol:n { ##1 }
5004         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5005           \l_stex_get_symbol_uri_str
5006         }
5007       }
5008     }
5009   }
5010 }

```

```

5008 }
5009 \exp_args:Nnnx
5010 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
5011 \str_if_empty:NF \sexamplotype {
5012   \stex_annotate_invisible:nnn{type}{\sexamplotype}{}
5013 }
5014 \clist_set:No \l_tmpa_clist \sexamplotype
5015 \tl_clear:N \l_tmpa_tl
5016 \clist_map_inline:Nn \l_tmpa_clist {
5017   \tl_if_exist:cT {__stex_statements_sexample_##1_start:}{
5018     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
5019   }
5020 }
5021 \tl_if_empty:NTF \l_tmpa_tl {
5022   \__stex_statements_sexample_start:
5023 }{
5024   \l_tmpa_tl
5025 }
5026 }
5027 \str_if_empty:NF \sexampleid {
5028   \stex_ref_new_doc_target:n \sexampleid
5029 }
5030 \stex_smsmode_do:
5031 }{
5032   \str_if_empty:NF \sexamplename { \stex_symdecl_do:nn{}{\sexamplename} }
5033   \stex_if_smsmode:F {
5034     \clist_set:No \l_tmpa_clist \sexamplotype
5035     \tl_clear:N \l_tmpa_tl
5036     \clist_map_inline:Nn \l_tmpa_clist {
5037       \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
5038         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
5039       }
5040     }
5041     \tl_if_empty:NTF \l_tmpa_tl {
5042       \__stex_statements_sexample_end:
5043     }{
5044       \l_tmpa_tl
5045     }
5046     \end{stex_annotate_env}
5047   }
5048 }

```

\stexpatchexample

```

5049
5050 \cs_new_protected:Nn \__stex_statements_sexample_start: {
5051   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplename {
5052     (\sexamplename)
5053   }~}
5054 }
5055 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
5056
5057 \newcommand\stexpatchexample[3] [] {
5058   \str_set:Nx \l_tmpa_str{ #1 }
5059   \str_if_empty:NTF \l_tmpa_str {

```

```

5060     \tl_set:Nn \__stex_statements_sexample_start: { #2 }
5061     \tl_set:Nn \__stex_statements_sexample_end: { #3 }
5062   }{
5063     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
5064     \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
5065   }
5066 }

```

\inlineex inline:

```

5110 \stex_smsmode_do:
5111 }

```

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

32.4 Logical Paragraphs

sparagraph

```

5112 \keys_define:nn { stex / sparagraph} {
5113   id      .str_set_x:N = \sparagraphid ,
5114   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
5115   type    .str_set_x:N = \sparagraphtype ,
5116   for     .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5117   from    .tl_set:N    = \sparagraphfrom ,
5118   to      .tl_set:N    = \sparagraphto ,
5119   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
5120   name    .str_set:N   = \sparagraphname
5121 }
5122
5123 \cs_new_protected:Nn \stex_sparagraph_args:n {
5124   \tl_clear:N \l_stex_sparagraph_title_tl
5125   \tl_clear:N \sparagraphfrom
5126   \tl_clear:N \sparagraphto
5127   \tl_clear:N \l_stex_sparagraph_start_tl
5128   \str_clear:N \sparagraphid
5129   \str_clear:N \sparagraphtype
5130   \clist_clear:N \l__stex_statements_sparagraph_for_clist
5131   \str_clear:N \sparagraphname
5132   \keys_set:nn { stex / sparagraph }{ #1 }
5133 }
5134 \newif\if@in@omtext\@in@omtextfalse
5135
5136 \NewDocumentEnvironment {sparagraph} { 0{} } {
5137   \stex_sparagraph_args:n { #1 }
5138   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5139     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
5140   }{
5141     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
5142   }
5143   \@in@omtexttrue
5144   \stex_if_smsmode:F {
5145     \seq_clear:N \l_tmpa_seq
5146     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5147       \tl_if_empty:nF{ ##1 }{
5148         \stex_get_symbol:n { ##1 }
5149         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5150           \l_stex_get_symbol_uri_str
5151         }
5152       }
5153     }
5154     \exp_args:Nnnx
5155     \begin{stex_annotate_env}{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}
5156     \str_if_empty:NF \sparagraphtype {

```

```

5157     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5158 }
5159 \str_if_empty:NF \sparagraphfrom {
5160     \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5161 }
5162 \str_if_empty:NF \sparagraphto {
5163     \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5164 }
5165 \clist_set:No \l_tmpa_clist \sparagraphtype
5166 \tl_clear:N \l_tmpa_tl
5167 \clist_map_inline:Nn \sparagraphtype {
5168     \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
5169         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
5170     }
5171 }
5172 \tl_if_empty:NTF \l_tmpa_tl {
5173     \__stex_statements_sparagraph_start:
5174 }{
5175     \l_tmpa_tl
5176 }
5177 }
5178 \clist_set:No \l_tmpa_clist \sparagraphtype
5179 \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}
5180 {
5181     \stex_reactivate_macro:N \definiendum
5182     \stex_reactivate_macro:N \definame
5183     \stex_reactivate_macro:N \Definame
5184     \stex_reactivate_macro:N \premise
5185     \stex_reactivate_macro:N \definens
5186 }
5187 \str_if_empty:NTF \sparagraphid {
5188     \str_if_empty:NTF \sparagraphname {
5189         \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5190             \stex_ref_new_doc_target:n {}
5191         }
5192     } {
5193         \stex_ref_new_doc_target:n {}
5194     }
5195 } {
5196     \stex_ref_new_doc_target:n \sparagraphid
5197 }
5198 \exp_args:NNx
5199 \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{symdoc}}{
5200     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5201         \tl_if_empty:nF{ ##1 }{
5202             \stex_get_symbol:n { ##1 }
5203             \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
5204         }
5205     }
5206 }
5207 \stex_smsmode_do:
5208 \ignorespacesandpars
5209 }{
5210     \str_if_empty:NF \sparagraphname {

```



```

5211 \stex_symdecl_do:nn{}{\sparagraphname}
5212 \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5213 }
5214 \stex_if_smsmode:F {
5215 \clist_set:N \l_tmpa_clist \sparagraphtype
5216 \tl_clear:N \l_tmpa_tl
5217 \clist_map_inline:Nn \l_tmpa_clist {
5218 \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
5219 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
5220 }
5221 }
5222 \tl_if_empty:NTF \l_tmpa_tl {
5223 \__stex_statements_sparagraph_end:
5224 }{
5225 \l_tmpa_tl
5226 }
5227 \end{stex_annotate_env}
5228 }
5229 }

```

\stexpatchparagraph

```

5230
5231 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
5232 \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
5233 \tl_if_empty:NF \l_stex_sparagraph_title_tl {
5234 \titleemph{\l_stex_sparagraph_title_tl}:~
5235 }
5236 }{
5237 \titleemph{\l_stex_sparagraph_start_tl}~
5238 }
5239 }
5240 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
5241
5242 \newcommand\stexpatchparagraph[3] [] {
5243 \str_set:Nx \l_tmpa_str{ #1 }
5244 \str_if_empty:NTF \l_tmpa_str {
5245 \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
5246 \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
5247 }{
5248 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2 }
5249 \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
5250 }
5251 }
5252
5253 \keys_define:nn { stex / inlinepara} {
5254 id .str_set_x:N = \sparagraphid ,
5255 type .str_set_x:N = \sparagraphtype ,
5256 for .clist_set:N = \l__stex_statements_sparagraph_for_clist ,
5257 from .tl_set:N = \sparagraphfrom ,
5258 to .tl_set:N = \sparagraphto ,
5259 name .str_set:N = \sparagraphname
5260 }
5261 \cs_new_protected:Nn \__stex_statements_inlinepara_args:n {
5262 \tl_clear:N \sparagraphfrom

```

```

5263 \tl_clear:N \sparagraphto
5264 \str_clear:N \sparagraphid
5265 \str_clear:N \sparagraphtype
5266 \clist_clear:N \l__stex_statements_sparagraph_for_clist
5267 \str_clear:N \sparagraphname
5268 \keys_set:nn { stex / inlinepara }{ #1 }
5269 }
5270 \NewDocumentCommand \inlinepara { 0{} m } {
5271   \beginingroup
5272   \__stex_statements_inlinepara_args:n{ #1 }
5273   \clist_set:No \l_tmpa_clist \sparagraphtype
5274   \str_if_empty:NTF \sparagraphid {
5275     \str_if_empty:NTF \sparagraphname {
5276       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5277         \stex_ref_new_doc_target:n {}
5278       }
5279     } {
5280       \stex_ref_new_doc_target:n {}
5281     }
5282   } {
5283     \stex_ref_new_doc_target:n \sparagraphid
5284   }
5285   \stex_if_smsmode:TF{
5286     \str_if_empty:NF \sparagraphname {
5287       \stex_symdecl_do:nn{}{\sparagraphname}
5288       \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5289     }
5290   }{
5291     \seq_clear:N \l_tmpa_seq
5292     \clist_map_inline:Nn \l__stex_statements_sparagraph_for_clist {
5293       \tl_if_empty:nF{ ##1 }{
5294         \stex_get_symbol:n { ##1 }
5295         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5296           \l_stex_get_symbol_uri_str
5297         }
5298       }
5299     }
5300     \exp_args:NNx
5301     \stex_annotate:nnn{paragraph}{\seq_use:Nn \l_tmpa_seq {,}}{
5302       \str_if_empty:NF \sparagraphtype {
5303         \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
5304       }
5305       \str_if_empty:NF \sparagraphfrom {
5306         \stex_annotate_invisible:nnn{from}{\sparagraphfrom}{}
5307       }
5308       \str_if_empty:NF \sparagraphto {
5309         \stex_annotate_invisible:nnn{to}{\sparagraphto}{}
5310       }
5311       \str_if_empty:NF \sparagraphname {
5312         \stex_symdecl_do:nn{}{\sparagraphname}
5313         \stex_ref_new_sym_target:n {\l_stex_current_module_str ? \sparagraphname}
5314       }
5315       \exp_args:NNx \clist_if_in:NnT \l_tmpa_clist {\tl_to_str:n{syndoc}}{
5316         \clist_map_inline:Nn \l_tmpa_seq {

```

```

5317         \stex_ref_new_sym_target:n {##1}
5318     }
5319 }
5320 #2
5321 }
5322 }
5323 \endgroup
5324 \stex_smsmode_do:
5325 }
5326

```

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

```

5327 </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

```
5328 <*package>
5329 <@@=stex_sproof>
5330
5331 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
5332
```

33.2 Proofs

We first define some keys for the proof environment.

```
5333 \keys_define:nn { stex / spf } {
5334   id          .str_set_x:N = \spfid,
5335   for         .clist_set:N = \l__stex_sproof_spf_for_clist ,
5336   from        .tl_set:N    = \l__stex_sproof_spf_from_tl ,
5337   proofend    .tl_set:N    = \l__stex_sproof_spf_proofend_tl,
5338   type        .str_set_x:N = \spftype,
5339   title       .tl_set:N    = \spftitle,
5340   continues   .tl_set:N    = \l__stex_sproof_spf_continues_tl,
5341   functions   .tl_set:N    = \l__stex_sproof_spf_functions_tl,
5342   method      .tl_set:N    = \l__stex_sproof_spf_method_tl
5343 }
5344 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
5345   \str_clear:N \spfid
5346   \tl_clear:N \l__stex_sproof_spf_for_tl
5347   \tl_clear:N \l__stex_sproof_spf_from_tl
5348   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
5349   \str_clear:N \spftype
5350   \tl_clear:N \spftitle
5351   \tl_clear:N \l__stex_sproof_spf_continues_tl
5352   \tl_clear:N \l__stex_sproof_spf_functions_tl

```

⁸EdNOTE: need an implementation for L^AT_EX_ML

```

5353 \tl_clear:N \l__stex_sproof_spf_method_tl
5354 \bool_set_false:N \l__stex_sproof_inc_counter_bool
5355 \keys_set:nn { stex / spf }{ #1 }
5356 }

```

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

```

5357 \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.

```

5358 \intarray_new:Nn\l__stex_sproof_counter_intarray{50}
5359 \cs_new_protected:Npn \sproofnumber {
5360   \int_set:Nn \l_tmpa_int {1}
5361   \bool_while_do:nn {
5362     \int_compare_p:nNn {
5363       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5364     } > 0
5365   }{
5366     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int .
5367     \int_incr:N \l_tmpa_int
5368   }
5369 }
5370 \cs_new_protected:Npn \__stex_sproof_inc_counter: {
5371   \int_set:Nn \l_tmpa_int {1}
5372   \bool_while_do:nn {
5373     \int_compare_p:nNn {
5374       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5375     } > 0
5376   }{
5377     \int_incr:N \l_tmpa_int
5378   }
5379   \int_compare:nNnF \l_tmpa_int = 1 {
5380     \int_decr:N \l_tmpa_int
5381   }
5382   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int {
5383     \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int + 1

```

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

```

5384 }
5385 }
5386
5387 \cs_new_protected:Npn \__stex_sproof_add_counter: {
5388   \int_set:Nn \l_tmpa_int {1}
5389   \bool_while_do:nn {
5390     \int_compare_p:nNn {
5391       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5392     } > 0
5393   }{
5394     \int_incr:N \l_tmpa_int
5395   }
5396   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 1 }
5397 }
5398
5399 \cs_new_protected:Npn \__stex_sproof_remove_counter: {
5400   \int_set:Nn \l_tmpa_int {1}
5401   \bool_while_do:nn {
5402     \int_compare_p:nNn {
5403       \intarray_item:Nn \l__stex_sproof_counter_intarray \l_tmpa_int
5404     } > 0
5405   }{
5406     \int_incr:N \l_tmpa_int
5407   }
5408   \int_decr:N \l_tmpa_int
5409   \intarray_gset:Nnn \l__stex_sproof_counter_intarray \l_tmpa_int { 0 }
5410 }

```

\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

```

5411 \def\sproof@box{
5412   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
5413 }
5414 \def\sproofend{
5415   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
5416     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
5417   }
5418 }

```

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

spf@*@kw

```

5419 \def\spf@proofsketch@kw{Proof-Sketch}
5420 \def\spf@proof@kw{Proof}
5421 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

5422 \AddToHook{begindocument}{
5423   \ltx@ifpackageloaded{babel}{
5424     \makeatletter
5425     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5426     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5427       \input{sproof-ngerman.ldf}

```

```

5428 }
5429 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5430   \input{sproof-finnish.ldf}
5431 }
5432 \clist_if_in:NnT \l_tmpa_clist {french}{
5433   \input{sproof-french.ldf}
5434 }
5435 \clist_if_in:NnT \l_tmpa_clist {russian}{
5436   \input{sproof-russian.ldf}
5437 }
5438 \makeatother
5439 }{}
5440 }

```

spfsketch

```

5441 \newcommand\spfsketch[2] [] {
5442   \beginingroup
5443   \let \premise \stex_proof_premise:
5444   \__stex_sproof_spf_args:n{#1}
5445   \stex_if_smsmode:TF {
5446     \str_if_empty:NF \spfid {
5447       \stex_ref_new_doc_target:n \spfid
5448     }
5449   }{
5450     \seq_clear:N \l_tmpa_seq
5451     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5452       \tl_if_empty:nF{ ##1 }{
5453         \stex_get_symbol:n { ##1 }
5454         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5455           \l_stex_get_symbol_uri_str
5456         }
5457       }
5458     }
5459     \exp_args:Nnx
5460     \stex_annotate:nnn{proofsketch}{\seq_use:Nn \l_tmpa_seq {,}}{
5461       \str_if_empty:NF \spftype {
5462         \stex_annotate_invisible:nnn{type}{\spftype}{-}
5463       }
5464       \clist_set:Nn \l_tmpa_clist \spftype
5465       \tl_set:Nn \l_tmpa_tl {
5466         \titleemph{
5467           \tl_if_empty:NTF \spftitle {
5468             \spf@proofsketch@kw
5469           }{
5470             \spftitle
5471           }
5472         }::~
5473       }
5474       \clist_map_inline:Nn \l_tmpa_clist {
5475         \exp_args:Nn \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5476           \tl_clear:N \l_tmpa_tl
5477         }
5478       }
5479       \str_if_empty:NF \spfid {

```

```

5480         \stex_ref_new_doc_target:n \spfid
5481     }
5482     \l_tmpa_tl #2 \sproofend
5483 }
5484 }
5485 \endgroup
5486 \stex_smsmode_do:
5487 }
5488

```

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

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

```

5489 \newenvironment{spfeq}[2][]{
5490   \__stex_sproof_spf_args:n{#1}
5491   \let \premise \stex_proof_premise:
5492   \stex_if_smsmode:TF {
5493     \str_if_empty:NF \spfid {
5494       \stex_ref_new_doc_target:n \spfid
5495     }
5496   }{
5497     \seq_clear:N \l_tmpa_seq
5498     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5499       \tl_if_empty:NF{ ##1 }{
5500         \stex_get_symbol:n { ##1 }
5501         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5502           \l_stex_get_symbol_uri_str
5503         }
5504       }
5505     }
5506     \exp_args:Nnnx
5507     \begin{stex_annotate_env}{spfeq}{\seq_use:Nn \l_tmpa_seq {,}}
5508     \str_if_empty:NF \spftype {
5509       \stex_annotate_invisible:nnn{type}{\spftype}{ }
5510     }
5511
5512     \clist_set:No \l_tmpa_clist \spftype
5513     \tl_clear:N \l_tmpa_tl
5514     \clist_map_inline:Nn \l_tmpa_clist {
5515       \tl_if_exist:cT {__stex_sproof_spfeq_##1_start:}{
5516         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_start:}}
5517       }
5518       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5519         \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5520       }
5521     }
5522     \tl_if_empty:NTF \l_tmpa_tl {
5523       \__stex_sproof_spfeq_start:
5524     }{
5525       \l_tmpa_tl
5526     }{-#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


```

5527 \str_if_empty:NF \spfid {
5528 \stex_ref_new_doc_target:n \spfid
5529 }
5530 \begin{displaymath}\begin{array}{rc1l}
5531 }
5532 \stex_smsmode_do:
5533 }{
5534 \stex_if_smsmode:F {
5535 \end{array}\end{displaymath}
5536 \clist_set:No \l_tmpa_clist \spftype
5537 \tl_clear:N \l_tmpa_tl
5538 \clist_map_inline:Nn \l_tmpa_clist {
5539 \tl_if_exist:cT {__stex_sproof_spfeq_##1_end:}{
5540 \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_spfeq_##1_end:}}
5541 }
5542 }
5543 \tl_if_empty:NTF \l_tmpa_tl {
5544 \__stex_sproof_spfeq_end:
5545 }{
5546 \l_tmpa_tl
5547 }
5548 \end{stex_annotate_env}
5549 }
5550 }
5551
5552 \cs_new_protected:Nn \__stex_sproof_spfeq_start: {
5553 \titleemph{
5554 \tl_if_empty:NTF \spftitle {
5555 \spf@proof@kw
5556 }{
5557 \spftitle
5558 }
5559 }:
5560 }
5561 \cs_new_protected:Nn \__stex_sproof_spfeq_end: {\sproofend}
5562
5563 \newcommand\stexpatchspfeq[3] [] {
5564 \str_set:Nx \l_tmpa_str{ #1 }
5565 \str_if_empty:NTF \l_tmpa_str {
5566 \tl_set:Nn \__stex_sproof_spfeq_start: { #2 }
5567 \tl_set:Nn \__stex_sproof_spfeq_end: { #3 }
5568 }{
5569 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_start:\endcsname{ #2 }
5570 \exp_after:wN \tl_set:Nn \csname __stex_sproof_spfeq_#1_end:\endcsname{ #3 }
5571 }
5572 }
5573

```

(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.

```

5574 \newenvironment{sproof}[2] []{

```

```

5575 \let \premise \stex_proof_premise:
5576 \intarray_gzero:N \l__stex_sproof_counter_intarray
5577 \intarray_gset:Nnn \l__stex_sproof_counter_intarray 1 1
5578 \__stex_sproof_spf_args:n{#1}
5579 \stex_if_smsmode:TF {
5580   \str_if_empty:NF \spfid {
5581     \stex_ref_new_doc_target:n \spfid
5582   }
5583 }{
5584   \seq_clear:N \l_tmpa_seq
5585   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5586     \tl_if_empty:NF{ ##1 }{
5587       \stex_get_symbol:n { ##1 }
5588       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5589         \l_stex_get_symbol_uri_str
5590       }
5591     }
5592   }
5593   \exp_args:Nnnx
5594   \begin{stex_annotate_env}{sproof}{\seq_use:Nn \l_tmpa_seq {,}}
5595   \str_if_empty:NF \spftype {
5596     \stex_annotate_invisible:nnn{type}{\spftype}{ }
5597   }
5598
5599   \clist_set:No \l_tmpa_clist \spftype
5600   \tl_clear:N \l_tmpa_tl
5601   \clist_map_inline:Nn \l_tmpa_clist {
5602     \tl_if_exist:cT {__stex_sproof_sproof_##1_start:}{
5603       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_start:}}
5604     }
5605     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5606       \tl_set:Nn \l_tmpa_tl {\use:n{ }}
5607     }
5608   }
5609   \tl_if_empty:NTF \l_tmpa_tl {
5610     \__stex_sproof_sproof_start:
5611   }{
5612     \l_tmpa_tl
5613   }{~#2}
5614   \str_if_empty:NF \spfid {
5615     \stex_ref_new_doc_target:n \spfid
5616   }
5617   \begin{description}
5618 }
5619 \stex_smsmode_do:
5620 }{
5621   \stex_if_smsmode:F{
5622     \end{description}
5623     \clist_set:No \l_tmpa_clist \spftype
5624     \tl_clear:N \l_tmpa_tl
5625     \clist_map_inline:Nn \l_tmpa_clist {
5626       \tl_if_exist:cT {__stex_sproof_sproof_##1_end:}{
5627         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_sproof_sproof_##1_end:}}
5628       }

```

```

5629     }
5630     \tl_if_empty:NTF \l_tmpa_tl {
5631       \__stex_sproof_sproof_end:
5632     }{
5633       \l_tmpa_tl
5634     }
5635     \end{stex_annotate_env}
5636   }
5637 }
5638
5639 \cs_new_protected:Nn \__stex_sproof_sproof_start: {
5640   \par\noindent\titleemph{
5641     \tl_if_empty:NTF \spftype {
5642       \spf@proof@kw
5643     }{
5644       \spftype
5645     }
5646   }:
5647 }
5648 \cs_new_protected:Nn \__stex_sproof_sproof_end: {\sproofend}
5649
5650 \newcommand\stexpatchproof[3] [] {
5651   \str_set:Nx \l_tmpa_str{ #1 }
5652   \str_if_empty:NTF \l_tmpa_str {
5653     \tl_set:Nn \__stex_sproof_sproof_start: { #2 }
5654     \tl_set:Nn \__stex_sproof_sproof_end: { #3 }
5655   }{
5656     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_start:\endcsname{ #2 }
5657     \exp_after:wN \tl_set:Nn \csname __stex_sproof_sproof_#1_end:\endcsname{ #3 }
5658   }
5659 }

```

\spfidea

```

5660 \newcommand\spfidea[2] []{
5661   \__stex_sproof_spf_args:n{#1}
5662   \titleemph{
5663     \tl_if_empty:NTF \spftype {Proof~Idea}{
5664       \spftype
5665     }:
5666   }~#2
5667   \sproofend
5668 }

```

(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

```

5669 \newenvironment{spfstep}[1] []{
5670   \__stex_sproof_spf_args:n{#1}
5671   \stex_if_smsmode:TF {

```

```

5672 \str_if_empty:NF \spfid {
5673   \stex_ref_new_doc_target:n \spfid
5674 }
5675 }{
5676   \@in@omtexttrue
5677   \seq_clear:N \l_tmpa_seq
5678   \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5679     \tl_if_empty:NF{ ##1 }{
5680       \stex_get_symbol:n { ##1 }
5681       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5682         \l_stex_get_symbol_uri_str
5683       }
5684     }
5685   }
5686   \exp_args:Nnnx
5687   \begin{stex_annotate_env}{spfstep}{\seq_use:Nn \l_tmpa_seq {,}}
5688   \str_if_empty:NF \spftype {
5689     \stex_annotate_invisible:nnn{type}{\spftype}{}
5690   }
5691   \clist_set:No \l_tmpa_clist \spftype
5692   \tl_set:Nn \l_tmpa_tl {
5693     \item[\sproofnumber]
5694     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5695   }
5696   \clist_map_inline:Nn \l_tmpa_clist {
5697     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5698       \tl_clear:N \l_tmpa_tl
5699     }
5700   }
5701   \l_tmpa_tl
5702   \tl_if_empty:NF \spftitle {
5703     {(\titleemph{\spftitle})\enspace}
5704   }
5705   \str_if_empty:NF \spfid {
5706     \stex_ref_new_doc_target:n \spfid
5707   }
5708 }
5709 \stex_smsmode_do:
5710 \ignorespacesandpars
5711 }{
5712   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5713     \__stex_sproof_inc_counter:
5714   }
5715   \stex_if_smsmode:F {
5716     \end{stex_annotate_env}
5717   }
5718 }

```

sproofcomment

```

5719 \newenvironment{sproofcomment}[1][]{
5720   \__stex_sproof_spf_args:n{#1}
5721   \clist_set:No \l_tmpa_clist \spftype
5722   \tl_set:Nn \l_tmpa_tl {
5723     \item[\sproofnumber]

```

```

5724 \bool_set_true:N \l__stex_sproof_inc_counter_bool
5725 }
5726 \clist_map_inline:Nn \l_tmpa_clist {
5727   \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5728     \tl_clear:N \l_tmpa_tl
5729   }
5730 }
5731 \l_tmpa_tl
5732 }{
5733   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5734     \__stex_sproof_inc_counter:
5735   }
5736 }

```

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.

```

5737 \newenvironment{subproof}[2][]{
5738   \__stex_sproof_spf_args:n{#1}
5739   \stex_if_smsmode:TF{
5740     \str_if_empty:NF \spfid {
5741       \stex_ref_new_doc_target:n \spfid
5742     }
5743   }{
5744     \seq_clear:N \l_tmpa_seq
5745     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5746       \tl_if_empty:nF{ ##1 }{
5747         \stex_get_symbol:n { ##1 }
5748         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5749           \l_stex_get_symbol_uri_str
5750         }
5751       }
5752     }
5753     \exp_args:Nnnx
5754     \begin{stex_annotate_env}{subproof}{\seq_use:Nn \l_tmpa_seq {,}}
5755     \str_if_empty:NF \spftype {
5756       \stex_annotate_invisible:nnn{type}{\spftype}{}
5757     }
5758
5759     \clist_set:No \l_tmpa_clist \spftype
5760     \tl_set:Nn \l_tmpa_tl {
5761       \item[\sproofnumber]
5762       \bool_set_true:N \l__stex_sproof_inc_counter_bool
5763     }
5764     \clist_map_inline:Nn \l_tmpa_clist {
5765       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5766         \tl_clear:N \l_tmpa_tl
5767       }
5768     }
5769     \l_tmpa_tl
5770     \tl_if_empty:NF \spftitle {
5771       {(\titleemph{\spftitle})\enspace}
5772     }

```

```

5773     {~#2}
5774     \str_if_empty:NF \spfid {
5775       \stex_ref_new_doc_target:n \spfid
5776     }
5777   }
5778   \__stex_sproof_add_counter:
5779   \stex_smsmode_do:
5780 }{
5781   \__stex_sproof_remove_counter:
5782   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5783     \__stex_sproof_inc_counter:
5784   }
5785   \stex_if_smsmode:F{
5786     \end{stex_annotate_env}
5787   }
5788 }

```

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

```

5789 \newenvironment{spfcases}[2][]{
5790   \tl_if_empty:nTF{#1}{
5791     \begin{subproof}[method=by-cases]{#2}
5792   }{
5793     \begin{subproof}[#1,method=by-cases]{#2}
5794   }
5795 }{
5796   \end{subproof}
5797 }

```

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

```

5798 \newenvironment{spfcase}[2][]{
5799   \__stex_sproof_spf_args:n{#1}
5800   \stex_if_smsmode:TF {
5801     \str_if_empty:NF \spfid {
5802       \stex_ref_new_doc_target:n \spfid
5803     }
5804   }{
5805     \seq_clear:N \l_tmpa_seq
5806     \clist_map_inline:Nn \l__stex_sproof_spf_for_clist {
5807       \tl_if_empty:nF{ ##1 }{
5808         \stex_get_symbol:n { ##1 }
5809         \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
5810           \l_stex_get_symbol_uri_str
5811         }
5812       }
5813     }
5814     \exp_args:Nnnx
5815     \begin{stex_annotate_env}{spfcase}{\seq_use:Nn \l_tmpa_seq {,}}
5816     \str_if_empty:NF \spftype {
5817       \stex_annotate_invisible:nnn{type}{\spftype}{}}
5818   }
5819   \clist_set:Nn \l_tmpa_clist \spftype
5820   \tl_set:Nn \l_tmpa_tl {
5821     \item[\sproofnumber]

```

```

5822     \bool_set_true:N \l__stex_sproof_inc_counter_bool
5823   }
5824   \clist_map_inline:Nn \l_tmpa_clist {
5825     \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5826       \tl_clear:N \l_tmpa_tl
5827     }
5828   }
5829   \l_tmpa_tl
5830   \tl_if_empty:nF{#2}{
5831     \titleemph{#2}:~
5832   }
5833 }
5834 \__stex_sproof_add_counter:
5835 \stex_smsmode_do:
5836 ){
5837   \__stex_sproof_remove_counter:
5838   \bool_if:NT \l__stex_sproof_inc_counter_bool {
5839     \__stex_sproof_inc_counter:
5840   }
5841   \stex_if_smsmode:F{
5842     \clist_set:No \l_tmpa_clist \spftype
5843     \tl_set:Nn \l_tmpa_tl{\sproofend}
5844     \clist_map_inline:Nn \l_tmpa_clist {
5845       \exp_args:No \str_if_eq:nnT \c__stex_sproof_flow_str {##1} {
5846         \tl_clear:N \l_tmpa_tl
5847       }
5848     }
5849     \l_tmpa_tl
5850     \end{stex_annotate_env}
5851   }
5852 }

```

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

```

5853 \newcommand\spfcasesketch[3][]{
5854   \begin{spfcase}[#1]{#2}#3\end{spfcase}
5855 }

```

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.

```

5856 \keys_define:nn { stex / just }{
5857   id      .str_set:x:N = \l__stex_sproof_just_id_str,
5858   method  .tl_set:N    = \l__stex_sproof_just_method_tl,
5859   premises .tl_set:N    = \l__stex_sproof_just_premises_tl,
5860   args     .tl_set:N    = \l__stex_sproof_just_args_tl
5861 }

```

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

```
5862 \newenvironment{justification}[1] [] {}{}
```

\premise

```
5863 \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.

```
5864 \newcommand\justarg[2] [] {#2}
```

```
5865 \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

```
5866 <*package>
5867
5868 %%%%%%%%%% others.dtx %%%%%%%%%%
5869
5870 <@@=stex_others>
    Warnings and error messages
5871 % None

\MSC Math subject classifier

5872 \NewDocumentCommand \MSC {m} {
5873 % TODO
5874 }

(End definition for \MSC. This function is documented on page ??.)
    Patching tikzinput, if loaded
5875 \@ifpackageloaded{tikzinput}{
5876 \RequirePackage{stex-tikzinput}
5877 }{}
5878 </package>
```

Chapter 35

STEX -Metatheory Implementation

```
5879 <*package>
5880 <@@=stex_modules>
5881
5882 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
5883
5884 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
5885 \begingroup
5886 \stex_module_setup:nn{
5887   ns=\c_stex_metatheory_ns_str,
5888   meta=NONE
5889 }{Metatheory}
5890 \stex_reactivate_macro:N \symdecl
5891 \stex_reactivate_macro:N \notation
5892 \stex_reactivate_macro:N \symdef
5893 \ExplSyntaxOff
5894 \csname stex_suppress_html:n\endcsname{
5895   % is-a (a:A, a \in A, a is an A, etc.)
5896   \symdecl{isa}[args=ai]
5897   \notation{isa}[typed,op=:]{#1 \comp{:} #2}{##1 \comp, ##2}
5898   \notation{isa}[in]{#1 \comp\in #2}{##1 \comp, ##2}
5899   \notation{isa}[pred]{#2\comp(#1 \comp)}{##1 \comp, ##2}
5900
5901   % bind (\forall, \Pi, \lambda etc.)
5902   \symdecl{bind}[args=Bi]
5903   \notation{bind}[forall]{\comp\forall #1.;#2}{##1 \comp, ##2}
5904   \notation{bind}[Pi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5905   \notation{bind}[deffun]{\comp( #1 \comp{ }\;\to\; ) #2}{##1 \comp, ##2}
5906
5907   % implicit bind
5908   \symdef{implicitbind}[args=Bi]{\comp\prod_{#1}#2}{##1 \comp, ##2}
5909
5910   % dummy variable
5911   \symdecl{dummyvar}
5912   \notation{dummyvar}[underscore]{\comp\_}
5913   \notation{dummyvar}[dot]{\comp\cdot}
```

```

5914 \notation{dummyvar}[dash]{\comp{\rm --}}
5915
5916 %fromto (function space, Hom-set, implication etc.)
5917 \symdecl{fromto}[args=ai]
5918 \notation{fromto}[xarrow]{#1 \comp\to #2}{##1 \comp\times ##2}
5919 \notation{fromto}[arrow]{#1 \comp\to #2}{##1 \comp\to ##2}
5920
5921 % mapto (lambda etc.)
5922 \symdecl{mapto}[args=Bi]
5923 \notation{mapto}[mapsto]{#1 \comp\mapsto #2}{#1 \comp, #2}
5924 \notation{mapto}[lambda]{\comp\lambda #1 \comp.; #2}{#1 \comp, #2}
5925 \notation{mapto}[lambdau]{\comp\lambda_{#1} \comp.; #2}{#1 \comp, #2}
5926
5927 % function/operator application
5928 \symdecl{apply}[args=ia]
5929 \notation{apply}[prec=0;0x\infpres,parens]{#1 \comp( #2 \comp)}{##1 \comp, ##2}
5930 \notation{apply}[prec=0;0x\infpres,lambda]{#1 \; #2 }{##1 \; ; ##2}
5931
5932 % ‘type’ of all collections (sets,classes,types,kinds)
5933 \symdecl{metacollection}
5934 \notation{metacollection}[U]{\comp{\mathcal{U}}}
5935 \notation{metacollection}[set]{\comp{\textsf{Set}}}
5936
5937 % collection of propositions/booleans/truth values
5938 \symdecl{prop}[name=proposition]
5939 \notation{prop}[prop]{\comp{\rm prop}}
5940 \notation{prop}[B00L]{\comp{\rm B00L}}
5941
5942 % sequences
5943 \symdecl{seqtype}[args=1]
5944 \notation{seqtype}[kleene]{#1^{\comp\ast}}
5945
5946 \symdef{sequence-index}[args=2,li,prec=nobrackets]{#{#1}_{#2}}
5947 \notation{sequence-index}[ui,prec=nobrackets]{#{#1}^{\comp\ast}}
5948
5949 \symdef{aseqdots}[args=a,prec=nobrackets]{#1\comp{,\ellipses}}{##1\comp,##2}
5950 \symdef{aseqfromto}[args=ai,prec=nobrackets]{#1\comp{,\ellipses,}#2}{##1\comp,##2}
5951 \symdef{aseqfromtovia}[args=aii,prec=nobrackets]{#1\comp{,\ellipses,}#2\comp{,\ellipses,}#3}
5952
5953 % letin (‘let’, local definitions, variable substitution)
5954 \symdecl{letin}[args=bii]
5955 \notation{letin}[let]{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
5956 \notation{letin}[subst]{#3 \comp[ #1 \comp/ #2 \comp]}
5957 \notation{letin}[frac]{#3 \comp[ \frac{#2}{#1} \comp]}
5958
5959 % structures
5960 \symdecl*{module-type}[args=1]
5961 \notation{module-type}{\mathtt{MOD} #1}
5962 \symdecl{mathstruct}[name=mathematical-structure,args=a] % TODO
5963 \notation{mathstruct}[angle,prec=nobrackets]{\comp\angle #1 \comp\rangle}{##1 \comp, ##2}
5964
5965 }
5966 \ExplSyntaxOn
5967 \stex_add_to_current_module:n{

```

```

5968 \let\nappa\apply
5969 \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
5970 \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
5971 \def\livar{\csname sequence-index\endcsname[li]}
5972 \def\uivar{\csname sequence-index\endcsname[ui]}
5973 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
5974 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
5975 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
5976 }
5977 \__stex_modules_end_module:
5978 \endgroup
5979 </package>

```

Chapter 36

Tikzinput Implementation

```
5980 <*package>
5981
5982 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
5983
5984 \ProvidesExplPackage{tikzinput}{2022/02/26}{3.0.1}{tikzinput package}
5985 \RequirePackage{l3keys2e}
5986
5987 \keys_define:nn { tikzinput } {
5988   image .bool_set:N = \c_tikzinput_image_bool,
5989   image .default:n = false ,
5990   unknown .code:n = {}
5991 }
5992
5993 \ProcessKeysOptions { tikzinput }
5994
5995 \bool_if:NTF \c_tikzinput_image_bool {
5996   \RequirePackage{graphicx}
5997
5998   \providecommand\usetikzlibrary[]{}
5999   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
6000 }{
6001   \RequirePackage{tikz}
6002   \RequirePackage{standalone}
6003
6004   \newcommand \tikzinput [2] [] {
6005     \setkeys{Gin}{#1}
6006     \ifx \Gin@ewidth \Gin@exclamation
6007       \ifx \Gin@eheight \Gin@exclamation
6008         \input { #2 }
6009       \else
6010         \resizebox{!}{ \Gin@eheight }{
6011           \input { #2 }
6012         }
6013       \fi
6014     \else
6015       \ifx \Gin@eheight \Gin@exclamation
6016         \resizebox{ \Gin@ewidth }{!}{
6017           \input { #2 }
```

```

6018     }
6019     \else
6020     \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
6021     \input { #2 }
6022     }
6023     \fi
6024     \fi
6025   }
6026 }
6027
6028 \newcommand \ctikzinput [2] [] {
6029   \begin{center}
6030     \tikzinput [ #1 ] { #2 }
6031   \end{center}
6032 }
6033
6034 \@ifpackageloaded{stex}{
6035   \RequirePackage{stex-tikzinput}
6036 }{}
6037
6038 </package>
6039 <*stex>
6040 \ProvidesExplPackage{stex-tikzinput}{2022/02/26}{3.0.1}{stex-tikzinput}
6041 \RequirePackage{stex}
6042 \RequirePackage{tikzinput}
6043
6044 \newcommand\mhtikzinput [2] [] {%
6045   \def\Gin@mhrepos{}\setkeys{Gin}{ #1 }%
6046   \stex_in_repository:nn\Gin@mhrepos{
6047     \tikzinput [ #1 ] {\mhp{ ##1 } { #2 }}
6048   }
6049 }
6050 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [ #1 ] { #2 }\end{center}}
6051 </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.

```
6052 \*cls)
6053 \@@=document_structure)
6054 \ProvidesExplClass{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure Class}
6055 \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`

```
6056 \keys_define:nn{ document-structure / pkg }{
6057   class      .str_set_x:N = \c_document_structure_class_str,
6058   minimal    .bool_set:N = \c_document_structure_minimal_bool,
6059   report     .code:n      = {
6060     \ClassWarning{document-structure}{the option 'report' is deprecated, use 'class=report',
6061     \str_set:Nn \c_document_structure_class_str {report}
6062   },
6063   book       .code:n      = {
6064     \ClassWarning{document-structure}{the option 'book' is deprecated, use 'class=book', ins
6065     \str_set:Nn \c_document_structure_class_str {book}
6066   },
6067   bookpart   .code:n      = {
6068     \ClassWarning{document-structure}{the option 'bookpart' is deprecated, use 'class=book,t
6069     \str_set:Nn \c_document_structure_class_str {book}
6070     \str_set:Nn \c_document_structure_topsect_str {chapter}
6071   },
```

```

6072 docopt      .str_set_x:N = \c_document_structure_docopt_str,
6073 unknown     .code:n      = {
6074   \PassOptionsToPackage{ \CurrentOption }{ document-structure }
6075 }
6076 }
6077 \ProcessKeysOptions{ document-structure / pkg }
6078 \str_if_empty:NT \c_document_structure_class_str {
6079   \str_set:Nn \c_document_structure_class_str {article}
6080 }
6081 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
6082   {\c_document_structure_class_str}
6083

```

37.3 Beefing up the document environment

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

```

6084 \RequirePackage{document-structure}
6085 \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_EX_ML.¹²

```

6086 \keys_define:nn { document-structure / document }{
6087   id .str_set_x:N = \c_document_structure_document_id_str
6088 }
6089 \let\__document_structure_orig_document=\document
6090 \renewcommand{\document}[1][]{
6091   \keys_set:nn{ document-structure / document }{ #1 }
6092   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
6093   \__document_structure_orig_document
6094 }

```

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

```

6095 }
6096 \</cls>

```

37.4 Implementation: document-structure Package

```

6097 \<*package>
6098 \ProvidesExplPackage{document-structure}{2022/02/26}{3.0.1}{Modular Document Structure}
6099 \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


```

6100
6101 \keys_define:nn{ document-structure / pkg }{
6102   class      .str_set_x:N = \c_document_structure_class_str,
6103   topsect    .str_set_x:N = \c_document_structure_topsect_str,
6104   % showignores .bool_set:N = \c_document_structure_showignores_bool,
6105 }
6106 \ProcessKeysOptions{ document-structure / pkg }
6107 \str_if_empty:NT \c_document_structure_class_str {
6108   \str_set:Nn \c_document_structure_class_str {article}
6109 }
6110 \str_if_empty:NT \c_document_structure_topsect_str {
6111   \str_set:Nn \c_document_structure_topsect_str {section}
6112 }

```

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

```

6113 \RequirePackage{xspace}
6114 \RequirePackage{comment}
6115 \AddToHook{begindocument}{
6116   \ltx@ifpackageloaded{babel}{
6117     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6118     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6119       \makeatletter\input{document-structure-ngerman.ldf}\makeatother
6120     }
6121   }{}
6122 }

```

`\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.

```

6123 \int_new:N \l_document_structure_section_level_int
6124 \str_case:VnF \c_document_structure_topsect_str {
6125   {part}}{
6126     \int_set:Nn \l_document_structure_section_level_int {0}
6127   }
6128   {chapter}}{
6129     \int_set:Nn \l_document_structure_section_level_int {1}
6130   }
6131 }{
6132   \str_case:VnF \c_document_structure_class_str {
6133     {book}}{
6134       \int_set:Nn \l_document_structure_section_level_int {0}
6135     }
6136     {report}}{
6137       \int_set:Nn \l_document_structure_section_level_int {0}
6138     }
6139   }{
6140     \int_set:Nn \l_document_structure_section_level_int {2}
6141   }
6142 }

```

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

```
6143 \def\current@section@level{document}%
6144 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
6145 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
6146 \cs_new_protected:Npn \skipomgroup {
6147   \ifcase\l_document_structure_section_level_int
6148   \or\stepcounter{part}
6149   \or\stepcounter{chapter}
6150   \or\stepcounter{section}
6151   \or\stepcounter{subsection}
6152   \or\stepcounter{subsubsection}
6153   \or\stepcounter{paragraph}
6154   \or\stepcounter{subparagraph}
6155   \fi
6156 }
```

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

`blindfragment`

```
6157 \newcommand\at@begin@blindomgroup[1]{%
6158 \newenvironment{blindfragment}
6159 {
6160   \int_incr:N\l_document_structure_section_level_int
6161   \at@begin@blindomgroup\l_document_structure_section_level_int
6162 }{}}
```

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

```
6163 \newcommand\omgroup@nonum[2]{%
6164   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
6165   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
6166 }
```

(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.

```
6167 \newcommand\omgroup@num[2]{%
```

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

```

6168 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
6169   \@nameuse{#1}{#2}
6170 }{
6171   \cs_if_exist:NTF\rdfmata@sectioning{
6172     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
6173   }{
6174     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
6175   }
6176 }
6177 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
6178 }

```

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

sfragment

```

6179 \keys_define:nn { document-structure / omgroup }{
6180   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
6181   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
6182   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
6183   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
6184   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
6185   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
6186   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
6187   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
6188   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
6189   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
6190 }
6191 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
6192   \str_clear:N \l__document_structure_omgroup_id_str
6193   \str_clear:N \l__document_structure_omgroup_date_str
6194   \clist_clear:N \l__document_structure_omgroup_creators_clist
6195   \clist_clear:N \l__document_structure_omgroup_contributors_clist
6196   \tl_clear:N \l__document_structure_omgroup_srccite_tl
6197   \tl_clear:N \l__document_structure_omgroup_type_tl
6198   \tl_clear:N \l__document_structure_omgroup_short_tl
6199   \tl_clear:N \l__document_structure_omgroup_display_tl
6200   \tl_clear:N \l__document_structure_omgroup_intro_tl
6201   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
6202   \keys_set:nn { document-structure / omgroup } { #1 }
6203 }

```

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.

```

6204 \newif\if@mainmatter\@mainmattertrue
6205 \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.

```

6206 \keys_define:nn { document-structure / sectioning }{
6207   name .str_set_x:N = \l__document_structure_sect_name_str ,
6208   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
6209   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
6210   clear .default:n = {true} ,
6211   num .bool_set:N = \l__document_structure_sect_num_bool ,

```

```

6212   num      .default:n      = {true}
6213 }
6214 \cs_new_protected:Nn \__document_structure_sect_args:n {
6215   \str_clear:N \l__document_structure_sect_name_str
6216   \str_clear:N \l__document_structure_sect_ref_str
6217   \bool_set_false:N \l__document_structure_sect_clear_bool
6218   \bool_set_false:N \l__document_structure_sect_num_bool
6219   \keys_set:nn { document-structure / sectioning } { #1 }
6220 }
6221 \newcommand\omdoc@sectioning[3][]{
6222   \__document_structure_sect_args:n {#1}
6223   \let\omdoc@sect@name\l__document_structure_sect_name_str
6224   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
6225   \if@mainmatter% numbering not overridden by frontmatter, etc.
6226     \bool_if:NTF \l__document_structure_sect_num_bool {
6227       \omgroup@num{#2}{#3}
6228     }{
6229       \omgroup@nonum{#2}{#3}
6230     }
6231     \def\current@section@level{\omdoc@sect@name}
6232   \else
6233     \omgroup@nonum{#2}{#3}
6234   \fi
6235 }% 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.

```

6236 \newcommand\omgroup@redefine@addtocontents[1]{%
6237 %\edef\__document_structureimport{#1}%
6238 %\@for\@I:=\__document_structureimport\do{%
6239 %\edef\@path{\csname module@\@I @path\endcsname}%
6240 %\@ifundefined{tf@toc}\relax%
6241 %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
6242 %\ifx\hyper@anchor\undefined% hyperref.sty loaded?
6243 %\def\addcontentsline##1##2##3{%
6244 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6245 %\else% hyperref.sty not loaded
6246 %\def\addcontentsline##1##2##3{%
6247 %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
6248 %\fi
6249 }% 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.

```

6250 \newenvironment{sfragment}[2][]{% keys, title
6251 {
6252   \__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.

```

6253   \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
6254     \omgroup@redefine@addtocontents{
6255       \@ifundefined{module@id}\used@modules%

```

```

6256     %{\@ifundefined{module@}\module@id @path}{\used@modules}\module@id}
6257   }
6258 }

```

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

```

6259 \int_incr:N\l_document_structure_section_level_int
6260 \ifcase\l_document_structure_section_level_int
6261   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
6262   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
6263   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
6264   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
6265   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
6266   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
6267   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
6268 \fi
6269 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
6270 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6271   \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6272 }
6273 }% for customization
6274 {}

```

and finally, we localize the sections

```

6275 \newcommand\omdoc@part@kw{Part}
6276 \newcommand\omdoc@chapter@kw{Chapter}
6277 \newcommand\omdoc@section@kw{Section}
6278 \newcommand\omdoc@subsection@kw{Subsection}
6279 \newcommand\omdoc@subsubsection@kw{Subsubsection}
6280 \newcommand\omdoc@paragraph@kw{paragraph}
6281 \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`

```

6282 \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).

```

6283 \cs_if_exist:NTF\frontmatter{
6284   \let\__document_structure_orig_frontmatter\frontmatter
6285   \let\frontmatter\relax
6286 }{
6287   \tl_set:Nn\__document_structure_orig_frontmatter{
6288     \clearpage
6289     \@mainmatterfalse
6290     \pagenumbering{roman}

```

```

6291 }
6292 }
6293 \cs_if_exist:NTF\backmatter{
6294   \let\__document_structure_orig_backmatter\backmatter
6295   \let\backmatter\relax
6296 }{
6297   \tl_set:Nn\__document_structure_orig_backmatter{
6298     \clearpage
6299     \@mainmatterfalse
6300     \pagenumbering{roman}
6301   }
6302 }

```

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.

```

6303 \newenvironment{frontmatter}{
6304   \__document_structure_orig_frontmatter
6305 }{
6306   \cs_if_exist:NTF\mainmatter{
6307     \mainmatter
6308   }{
6309     \clearpage
6310     \@mainmattertrue
6311     \pagenumbering{arabic}
6312   }
6313 }

```

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

```

6314 \newenvironment{backmatter}{
6315   \__document_structure_orig_backmatter
6316 }{
6317   \cs_if_exist:NTF\mainmatter{
6318     \mainmatter
6319   }{
6320     \clearpage
6321     \@mainmattertrue
6322     \pagenumbering{arabic}
6323   }
6324 }

```

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

```

6325 \@mainmattertrue\pagenumbering{arabic}

```

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

```

6326 \def \c__document_structure_document_str{document}
6327 \newcommand\afterprematurestop{}
6328 \def\prematurestop@endomgroup{
6329   \unless\ifx\@currenvir\c__document_structure_document_str
6330     \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter\expandafter\expandafter
6331     \expandafter\prematurestop@endomgroup

```

```

6332 \fi
6333 }
6334 \providecommand\prematurestop{
6335   \message{Stopping~sTeX~processing~prematurely}
6336   \prematurestop@endgroup
6337   \afterprematurestop
6338   \end{document}
6339 }

```

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

37.8 Global Variables

\setSGvar set a global variable

```

6340 \RequirePackage{etoolbox}
6341 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

6342 \newrobustcmd\useSGvar[1]{%
6343   \@ifundefined{sTeX@Gvar@#1}
6344   {\PackageError{document-structure}
6345    {The sTeX Global variable #1 is undefined}
6346    {set it with \protect\setSGvar}}
6347   \@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.

```

6348 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
6349   \@ifundefined{sTeX@Gvar@#1}
6350   {\PackageError{document-structure}
6351    {The sTeX Global variable #1 is undefined}
6352    {set it with \protect\setSGvar}}
6353   {\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.

```
6354 \*cls)
6355 \@@=notesslides)
6356 \ProvidesExplClass{notesslides}{2022/02/28}{3.1.0}{notesslides Class}
6357 \RequirePackage{13keys2e}
6358
6359 \keys_define:nn{notesslides / cls}{
6360   class .code:n = {
6361     \PassOptionsToClass{\CurrentOption}{document-structure}
6362     \str_if_eq:nnT{#1}{book}{
6363       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6364     }
6365     \str_if_eq:nnT{#1}{report}{
6366       \PassOptionsToPackage{defaulttopsec=part}{notesslides}
6367     }
6368   },
6369   notes .bool_set:N = \c__notesslides_notes_bool ,
6370   slides .code:n = { \bool_set_false:N \c__notesslides_notes_bool },
6371   unknown .code:n = {
6372     \PassOptionsToClass{\CurrentOption}{document-structure}
6373     \PassOptionsToClass{\CurrentOption}{beamer}
6374     \PassOptionsToPackage{\CurrentOption}{notesslides}
6375   }
6376 }
6377 \ProcessKeysOptions{ notesslides / cls }
6378 \bool_if:NTF \c__notesslides_notes_bool {
6379   \PassOptionsToPackage{notes=true}{notesslides}
6380 }{
6381   \PassOptionsToPackage{notes=false}{notesslides}
6382 }
6383 \</cls)
```


now we do the same for the notesslides package.

```

6384 <*package>
6385 \ProvidesExplPackage{notesslides}{2022/02/28}{3.1.0}{notesslides Package}
6386 \RequirePackage{13keys2e}
6387
6388 \keys_define:nn{notesslides / pkg}{
6389   topsect      .str_set_x:N = \c_notesslides_topsect_str,
6390   defaulttopsect .str_set_x:N = \c_notesslides_defaulttopsec_str,
6391   notes        .bool_set:N = \c_notesslides_notes_bool ,
6392   slides       .code:n      = { \bool_set_false:N \c_notesslides_notes_bool },
6393   sectocframes .bool_set:N = \c_notesslides_sectocframes_bool ,
6394   frameimages  .bool_set:N = \c_notesslides_frameimages_bool ,
6395   fiboxed      .bool_set:N = \c_notesslides_fiboxed_bool ,
6396   nopproblems  .bool_set:N = \c_notesslides_nopproblems_bool,
6397   unknown      .code:n      = {
6398     \PassOptionsToClass{\CurrentOption}{stex}
6399     \PassOptionsToClass{\CurrentOption}{tikzinput}
6400   }
6401 }
6402 \ProcessKeysOptions{ notesslides / pkg }
6403 \newif\ifnotes
6404 \bool_if:NTF \c_notesslides_notes_bool {
6405   \notesttrue
6406 }{
6407   \notesfalse
6408 }
6409

```

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

```

6410 \str_if_empty:NTF \c_notesslides_topsect_str {
6411   \str_set_eq:NN \__notesslides_topsect \c_notesslides_defaulttopsec_str
6412 }{
6413   \str_set_eq:NN \__notesslides_topsect \c_notesslides_topsect_str
6414 }
6415 </package>

```

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

```

6416 <*cls>
6417 \bool_if:NTF \c_notesslides_notes_bool {
6418   \LoadClass{document-structure}
6419 }{
6420   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
6421   \newcounter{Item}
6422   \newcounter{paragraph}
6423   \newcounter{subparagraph}
6424   \newcounter{Hfootnote}
6425   \RequirePackage{document-structure}
6426 }

```

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

```

6427 \RequirePackage{notesslides}
6428 </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.

```

6429 <*package>
6430 \bool_if:NT \c__notesslides_notes_bool {
6431   \RequirePackage{a4wide}
6432   \RequirePackage{marginnote}
6433   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
6434   \RequirePackage{mdframed}
6435   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
6436   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
6437 }
6438 \RequirePackage{stex-tikzinput}
6439 \RequirePackage{etoolbox}
6440 \RequirePackage{amssymb}
6441 \RequirePackage{amsmath}
6442 \RequirePackage{comment}
6443 \RequirePackage{textcomp}
6444 \RequirePackage{url}
6445 \RequirePackage{graphicx}
6446 \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

```

6447 \bool_if:NT \c__notesslides_notes_bool {
6448   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
6449 }
6450
6451
6452 \NewDocumentCommand \libusetheme {0{} m} {
6453   \bool_if:NTF \c__notesslides_notes_bool {
6454     \libusepackage[#1]{beamernotestheme#2}
6455   }{
6456     \libusepackage[#1]{beamertheme#2}
6457   }
6458 }

```

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

```

6459 \newcounter{slide}
6460 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
6461 \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.

```

6462 \bool_if:NTF \c__notesslides_notes_bool {
6463   \renewenvironment{note}{\ignorespaces}{}
6464 }{
6465   \excludecomment{note}
6466 }

```

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.

```

6467 \bool_if:NT \c__notesslides_notes_bool {
6468   \newlength{\slideframewidth}
6469   \setlength{\slideframewidth}{1.5pt}

```

frame We first define the keys.

```

6470 \cs_new_protected:Nn \__notesslides_do_yes_param:Nn {
6471   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
6472     \bool_set_true:N #1
6473   }{
6474     \bool_set_false:N #1
6475   }
6476 }
6477 \keys_define:nn{notesslides / frame}{
6478   label .str_set_x:N = \l__notesslides_frame_label_str,
6479   allowframebreaks .code:n = {
6480     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowframebreaks_bool { #1 }
6481   },
6482   allowdisplaybreaks .code:n = {
6483     \__notesslides_do_yes_param:Nn \l__notesslides_frame_allowdisplaybreaks_bool { #1 }
6484   },
6485   fragile .code:n = {
6486     \__notesslides_do_yes_param:Nn \l__notesslides_frame_fragile_bool { #1 }
6487   },
6488   shrink .code:n = {
6489     \__notesslides_do_yes_param:Nn \l__notesslides_frame_shrink_bool { #1 }
6490   },
6491   squeeze .code:n = {
6492     \__notesslides_do_yes_param:Nn \l__notesslides_frame_squeeze_bool { #1 }
6493   },
6494   t .code:n = {
6495     \__notesslides_do_yes_param:Nn \l__notesslides_frame_t_bool { #1 }
6496   },
6497 }
6498 \cs_new_protected:Nn \__notesslides_frame_args:n {
6499   \str_clear:N \l__notesslides_frame_label_str
6500   \bool_set_true:N \l__notesslides_frame_allowframebreaks_bool
6501   \bool_set_true:N \l__notesslides_frame_allowdisplaybreaks_bool
6502   \bool_set_true:N \l__notesslides_frame_fragile_bool
6503   \bool_set_true:N \l__notesslides_frame_shrink_bool
6504   \bool_set_true:N \l__notesslides_frame_squeeze_bool
6505   \bool_set_true:N \l__notesslides_frame_t_bool

```

```

6506     \keys_set:nn { notesslides / frame }{ #1 }
6507 }

```

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

```

6508 \renewenvironment{frame}[1][]{
6509     \__notesslides_frame_args:n{#1}
6510     \sffamily
6511     \stepcounter{slide}
6512     \def\@currentlabel{\theslide}
6513     \str_if_empty:NF \l__notesslides_frame_label_str {
6514         \label{\l__notesslides_frame_label_str}
6515     }

```

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

```

6516     \def\itemize@level{outer}
6517     \def\itemize@outer{outer}
6518     \def\itemize@inner{inner}
6519     \renewcommand\newpage{\addtocounter{framenumber}{1}}
6520     \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
6521     \renewenvironment{itemize}{
6522         \ifx\itemize@level\itemize@outer
6523             \def\itemize@label{$\rhd$}
6524         \fi
6525         \ifx\itemize@level\itemize@inner
6526             \def\itemize@label{$\scriptstyle\rhd$}
6527         \fi
6528         \begin{list}
6529             {\itemize@label}
6530             {\setlength{\labelsep}{.3em}
6531              \setlength{\labelwidth}{.5em}
6532              \setlength{\leftmargin}{1.5em}
6533             }
6534         \edef\itemize@level{\itemize@inner}
6535     }{
6536         \end{list}
6537     }

```

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

```

6538     \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
6539     }{
6540         \medskip\miko@slidelabel\end{mdframed}
6541     }

```

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

`\frametitle`

```

6542     \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
6543 }

```

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

EdN:15

`\pause`

```

15
6544 \bool_if:NT \c__notesslides_notes_bool {
6545     \newcommand\pause{}
6546 }

```

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

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

nparagraph

```
6547 \bool_if:NTF \c__notesslides_notes_bool {  
6548   \newenvironment{nparagraph}[1] [] {\begin{sparagraph}[#1]}\end{sparagraph}}  
6549 }{  
6550   \excludecomment{nparagraph}  
6551 }
```

nfragment

```
6552 \bool_if:NTF \c__notesslides_notes_bool {  
6553   \newenvironment{nfragment}[2] [] {\begin{sfragment}[#1]{#2}}\end{sfragment}}  
6554 }{  
6555   \excludecomment{nfragment}  
6556 }
```

ndefinition

```
6557 \bool_if:NTF \c__notesslides_notes_bool {  
6558   \newenvironment{ndefinition}[1] [] {\begin{sdefinition}[#1]}\end{sdefinition}}  
6559 }{  
6560   \excludecomment{ndefinition}  
6561 }
```

nassertion

```
6562 \bool_if:NTF \c__notesslides_notes_bool {  
6563   \newenvironment{nassertion}[1] [] {\begin{sassertion}[#1]}\end{sassertion}}  
6564 }{  
6565   \excludecomment{nassertion}  
6566 }
```

nsproof

```
6567 \bool_if:NTF \c__notesslides_notes_bool {  
6568   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}\end{sproof}}  
6569 }{  
6570   \excludecomment{nsproof}  
6571 }
```

nexample

```
6572 \bool_if:NTF \c__notesslides_notes_bool {  
6573   \newenvironment{nexample}[1] [] {\begin{sexample}[#1]}\end{sexample}}  
6574 }{  
6575   \excludecomment{nexample}  
6576 }
```

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

```
6577 \def\inputref@preskip{\smallskip}  
6578 \def\inputref@postskip{\medskip}
```

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

`\inputref*`

```
6579 \let\orig@inputref\inputref
6580 \def\inputref{\@ifstar\ninputref\orig@inputref}
6581 \newcommand\ninputref[2][] {
6582   \bool_if:NT \c__notesslides_notes_bool {
6583     \orig@inputref[#1]{#2}
6584   }
6585 }
```

(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>}`.

```
6586 \newlength{\slidelogoheight}
6587
6588 \bool_if:NTF \c__notesslides_notes_bool {
6589   \setlength{\slidelogoheight}{.4cm}
6590 }{
6591   \setlength{\slidelogoheight}{1cm}
6592 }
6593 \newsavebox{\slidelogo}
6594 \sbox{\slidelogo}{\sTeX}
6595 \newrobustcmd{\setslidelogo}[1]{
6596   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
6597 }
```

(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.

```
6598 \def\source{Michael Kohlhase}% customize locally
6599 \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.

```
6600 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
6601 \newsavebox{\cclogo}
6602 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{stex-cc_somerights}}
6603 \newif\ifcchref\cchreffalse
6604 \AtBeginDocument{
6605   \@ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
6606 }
6607 \def\licensing{
6608   \ifcchref
```

```

6609     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
6610   \else
6611     {\usebox{\cclogo}}
6612   \fi
6613 }
6614 \newrobustcmd{\setlicensing}[2][]{
6615   \def\@url{#1}
6616   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
6617   \ifx\@url\@empty
6618     \def\licensing{\usebox{\cclogo}}
6619   \else
6620     \def\licensing{
6621       \ifcchref
6622         \href{#1}{\usebox{\cclogo}}
6623       \else
6624         {\usebox{\cclogo}}
6625       \fi
6626     }
6627   \fi
6628 }

```

(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

```

6629 \newrobustcmd\miko@slidelabel{
6630   \vbox to \slidelogoheight{
6631     \vss\hbox to \slidewidth
6632     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
6633   }
6634 }

```

(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.

```

6635 \def\Gin@mhrepos{}
6636 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
6637 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
6638 \newrobustcmd\frameimage[2][]{
6639   \stepcounter{slide}
6640   \bool_if:NT \c__notesslides_frameimages_bool {
6641     \def\Gin@ewidth{}\setkeys{Gin}{#1}
6642     \bool_if:NF \c__notesslides_notes_bool { \vfill }
6643     \begin{center}
6644       \bool_if:NTF \c__notesslides_fiboxed_bool {
6645         \fbox{
6646           \ifx\Gin@ewidth\@empty
6647             \ifx\Gin@mhrepos\@empty
6648               \mhgraphics[width=\slidewidth,#1]{#2}
6649             \else

```

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

```

6650         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6651     \fi
6652 \else% Gin@ewidth empty
6653     \ifx\Gin@mhrepos\@empty
6654         \mhgraphics[#1]{#2}
6655     \else
6656         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6657     \fi
6658 \fi% Gin@ewidth empty
6659 }
6660 }{
6661     \ifx\Gin@ewidth\@empty
6662     \ifx\Gin@mhrepos\@empty
6663         \mhgraphics[width=\slidewidth,#1]{#2}
6664     \else
6665         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
6666     \fi
6667     \ifx\Gin@mhrepos\@empty
6668         \mhgraphics[#1]{#2}
6669     \else
6670         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
6671     \fi
6672 \fi% Gin@ewidth empty
6673 }
6674 \end{center}
6675 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
6676 \bool_if:NF \c__notesslides_notes_bool { \vfill }
6677 }
6678 } % 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.

```

6679 \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

```

6680 \AddToHook{begindocument}{
6681     \definecolor{green}{rgb}{0,.5,0}
6682     \definecolor{purple}{cmyk}{.3,1,0,.17}
6683 }

```

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`.

```

6684 % \def\STpresent#1{\textcolor{blue}{#1}}
6685 \def\defemph#1{\textcolor{magenta}{#1}}
6686 \def\symrefemph#1{\textcolor{cyan}{#1}}
6687 \def\compemph#1{\textcolor{blue}{#1}}
6688 \def\titleemph#1{\textcolor{blue}{#1}}
6689 \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.

```

6690 \pgfdeclareimage[width=.8em]{miko@small@dbend}{stex-dangerous-bend}
6691 \def\smalltextwarning{
6692   \pgfuseimage{miko@small@dbend}
6693   \xspace
6694 }
6695 \pgfdeclareimage[width=1.2em]{miko@dbend}{stex-dangerous-bend}
6696 \newrobustcmd\textwarning{
6697   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
6698   \xspace
6699 }
6700 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{stex-dangerous-bend}
6701 \newrobustcmd\bigtextwarning{
6702   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
6703   \xspace
6704 }

```

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

```

6705 \newrobustcmd\putgraphicsat[3]{
6706   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
6707 }
6708 \newrobustcmd\putat[2]{
6709   \begin{picture}(0,0)\put(#1){#2}\end{picture}
6710 }

```

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`.

```

6711 \bool_if:NT \c__notesslides_sectocframes_bool {
6712   \str_if_eq:VnTF \__notesslidesstopsect{part}{
6713     \newcounter{chapter}\counterwithin*{section}{chapter}
6714   }{
6715     \str_if_eq:VnT \__notesslidesstopsect{chapter}{
6716       \newcounter{chapter}\counterwithin*{section}{chapter}
6717     }
6718   }
6719 }

```

`\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
6720 \def\part@prefix{}
6721 \@ifpackageloaded{document-structure}{}{
6722   \str_case:VnF \__notesslidesstopsect {
6723     {part}{
6724       \int_set:Nn \l_document_structure_section_level_int {0}
6725       \def\thesection{\arabic{chapter}.\arabic{section}}

```

```

6726     \def\part@prefix{\arabic{chapter}.}
6727   }
6728   {chapter}{
6729     \int_set:Nn \l_document_structure_section_level_int {1}
6730     \def\thesection{\arabic{chapter}.\arabic{section}}
6731     \def\part@prefix{\arabic{chapter}.}
6732   }
6733   }{
6734     \int_set:Nn \l_document_structure_section_level_int {2}
6735     \def\part@prefix{}
6736   }
6737 }
6738
6739 \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

```

6740 \renewenvironment{sfragment}[2][]{
6741   \_document_structure_omgroup_args:n { #1 }
6742   \int_incr:N \l_document_structure_section_level_int
6743   \bool_if:NT \c__notesslides_sectocframes_bool {
6744     \stepcounter{slide}
6745     \begin{frame}[noframenumbering]
6746       \vfill\Large\centering
6747     \red{
6748       \ifcase\l_document_structure_section_level_int\or
6749         \stepcounter{part}
6750         \def\__notesslideslabel{\omdoc@part@kw~\Roman{part}}
6751         \def\currentsectionlevel{\omdoc@part@kw}
6752       \or
6753         \stepcounter{chapter}
6754         \def\__notesslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
6755         \def\currentsectionlevel{\omdoc@chapter@kw}
6756       \or
6757         \stepcounter{section}
6758         \def\__notesslideslabel{\part@prefix\arabic{section}}
6759         \def\currentsectionlevel{\omdoc@section@kw}
6760       \or
6761         \stepcounter{subsection}
6762         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
6763         \def\currentsectionlevel{\omdoc@subsection@kw}
6764       \or
6765         \stepcounter{subsubsection}
6766         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6767         \def\currentsectionlevel{\omdoc@subsubsection@kw}
6768       \or
6769         \stepcounter{paragraph}
6770         \def\__notesslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{s
6771         \def\currentsectionlevel{\omdoc@paragraph@kw}
6772       \else
6773         \def\__notesslideslabel{}

```

```

6774         \def\currentsectionlevel{\omdoc@paragraph@kw}
6775         \fi% end ifcase
6776         \_notesslideslabel%\sref@label@id\_notesslideslabel
6777         \quad #2%
6778     }%
6779     \vfill%
6780     \end{frame}%
6781 }
6782 \str_if_empty:NF \l__document_structure_omgroup_id_str {
6783     \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
6784 }
6785 }{}
6786 }

```

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

```

6787 \def\inserttheorembodyfont{\normalfont}
6788 %\bool_if:NF \c__notesslides_notes_bool {
6789 % \defbeamertemplate{theorem begin}{miko}
6790 % {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
6791 % \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
6792 % \inserttheorempunctuation\inserttheorembodyfont\xspace}
6793 % \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

6794 % \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.

```

6795 % \expandafter\def\csname Parent2\endcsname{}
6796 %}
6797
6798 \AddToHook{begindocument}{% this does not work for some reason
6799     \setbeamertemplate{theorems}[ams style]
6800 }
6801 \bool_if:NT \c__notesslides_notes_bool {
6802     \renewenvironment{columns}[1][{}]{%
6803         \par\noindent%
6804         \begin{minipage}%
6805             \slidewidth\centering\leavevmode%
6806     }{}%
6807     \end{minipage}\par\noindent%
6808 }%
6809 \newsavebox\columnbox%
6810 \renewenvironment<>{column}[2][{}]{%
6811     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
6812 }{}%
6813     \end{minipage}\end{lrbox}\usebox\columnbox%
6814 }%
6815 }
6816 \bool_if:NFT \c__notesslides_noproblems_bool {
6817     \newenvironment{problems}{}{}
6818 }{
6819     \excludecomment{problems}
6820 }

```

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.

```

6821 \gdef\printexcursions{}
6822 \newcommand\excursionref[2]{% label, text
6823   \bool_if:NT \c__notesslides_notes_bool {
6824     \begin{sparagraph}[title=Excursion]
6825       #2 \sref[fallback=the appendix]{#1}.
6826     \end{sparagraph}
6827   }
6828 }
6829 \newcommand\activate@excursion[2][]{
6830   \gappto\printexcursions{\inputref{#1}{#2}}
6831 }
6832 \newcommand\excursion[4][]{% repos, label, path, text
6833   \bool_if:NT \c__notesslides_notes_bool {
6834     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
6835   }
6836 }

```

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

`\excursiongroup`

```

6837 \keys_define:nn{notesslides / excursiongroup }{
6838   id          .str_set_x:N = \l__notesslides_excursion_id_str,
6839   intro       .tl_set:N   = \l__notesslides_excursion_intro_tl,
6840   mhrepos     .str_set_x:N = \l__notesslides_excursion_mhrepos_str
6841 }
6842 \cs_new_protected:Nn \__notesslides_excursion_args:n {
6843   \tl_clear:N \l__notesslides_excursion_intro_tl
6844   \str_clear:N \l__notesslides_excursion_id_str
6845   \str_clear:N \l__notesslides_excursion_mhrepos_str
6846   \keys_set:nn {notesslides / excursiongroup }{ #1 }
6847 }
6848 \newcommand\excursiongroup[1][]{
6849   \__notesslides_excursion_args:n{ #1 }
6850   \ifdefempty\printexcursions{}% only if there are excursions
6851   {\begin{note}
6852     \begin{sfragment}[#1]{Excursions}%
6853     \ifdefempty\l__notesslides_excursion_intro_tl{
6854       \inputref[\l__notesslides_excursion_mhrepos_str]{
6855         \l__notesslides_excursion_intro_tl
6856       }
6857     }
6858     \printexcursions%
6859     \end{sfragment}
6860     \end{note}}
6861 }
6862 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
6863 \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.

```
6864 <*package>
6865 <@@=problems>
6866 \ProvidesExplPackage{problem}{2022/02/26}{3.0.1}{Semantic Markup for Problems}
6867 \RequirePackage{l3keys2e,stex}
6868
6869 \keys_define:nn { problem / pkg }{
6870   notes      .default:n    = { true },
6871   notes      .bool_set:N   = \c__problems_notes_bool,
6872   gnotes     .default:n    = { true },
6873   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
6874   hints      .default:n    = { true },
6875   hints      .bool_set:N   = \c__problems_hints_bool,
6876   solutions  .default:n    = { true },
6877   solutions  .bool_set:N   = \c__problems_solutions_bool,
6878   pts        .default:n    = { true },
6879   pts        .bool_set:N   = \c__problems_pts_bool,
6880   min        .default:n    = { true },
6881   min        .bool_set:N   = \c__problems_min_bool,
6882   boxed      .default:n    = { true },
6883   boxed      .bool_set:N   = \c__problems_boxed_bool,
6884   unknown    .code:n       = {}
6885 }
6886 \newif\ifsolutions
6887
6888 \ProcessKeysOptions{ problem / pkg }
6889 \bool_if:NTF \c__problems_solutions_bool {
6890   \solutionstrue
6891 }{
6892   \solutionsfalse
6893 }
```

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

```
6894 \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.

```
6895 \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.

```
6896 \def\prob@problem@kw{Problem}
6897 \def\prob@solution@kw{Solution}
6898 \def\prob@hint@kw{Hint}
6899 \def\prob@note@kw{Note}
6900 \def\prob@gnote@kw{Grading}
6901 \def\prob@pt@kw{pt}
6902 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
6903 \AddToHook{begindocument}{
6904   \ltx@ifpackageloaded{babel}{
6905     \makeatletter
6906     \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
6907     \clist_if_in:NnT \l_tmpa_clist {ngerman}{
6908       \input{problem-ngerman.ldf}
6909     }
6910     \clist_if_in:NnT \l_tmpa_clist {finnish}{
6911       \input{problem-finnish.ldf}
6912     }
6913     \clist_if_in:NnT \l_tmpa_clist {french}{
6914       \input{problem-french.ldf}
6915     }
6916     \clist_if_in:NnT \l_tmpa_clist {russian}{
6917       \input{problem-russian.ldf}
6918     }
6919     \makeatother
6920   }{ }
6921 }
```

39.2 Problems and Solutions

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

```
6922 \keys_define:nn{ problem / problem }{
6923   id      .str_set:x:N = \l__problems_prob_id_str,
6924   pts     .tl_set:N    = \l__problems_prob_pts_tl,
6925   min     .tl_set:N    = \l__problems_prob_min_tl,
6926   title   .tl_set:N    = \l__problems_prob_title_tl,
6927   type    .tl_set:N    = \l__problems_prob_type_tl,
6928   refnum  .int_set:N    = \l__problems_prob_refnum_int
6929 }
6930 \cs_new_protected:Nn \__problems_prob_args:n {
```

```

6931 \str_clear:N \l__problems_prob_id_str
6932 \tl_clear:N \l__problems_prob_pts_tl
6933 \tl_clear:N \l__problems_prob_min_tl
6934 \tl_clear:N \l__problems_prob_title_tl
6935 \tl_clear:N \l__problems_prob_type_tl
6936 \int_zero_new:N \l__problems_prob_refnum_int
6937 \keys_set:nn { problem / problem }{ #1 }
6938 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
6939   \let\l__problems_prob_refnum_int\undefined
6940 }
6941 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

6942 \newcounter{problem}
6943 \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.

```

6944 \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

```

6945 \newcommand\prob@number{
6946   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
6947     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
6948   }{
6949     \int_if_exist:NTF \l__problems_prob_refnum_int {
6950       \prob@label{\int_use:N \l__problems_prob_refnum_int }
6951     }{
6952       \prob@label\theproblem
6953     }
6954   }
6955 }

```

(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.

```

6956 \newcommand\prob@title[3]{%
6957   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
6958     #2 \l__problems_inclprob_title_tl #3
6959   }{
6960     \tl_if_exist:NTF \l__problems_prob_title_tl {
6961       #2 \l__problems_prob_title_tl #3
6962     }{
6963       #1
6964     }
6965   }
6966 }

```

(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.

```

6967 \def\prob@heading{
6968   {\prob@problem@kw}\ \prob@number\prob@title{~}{~}{~}\strut}
6969   %\sref@label{id}\prob@problem@kw~\prob@number}{~}
6970 }

```

(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

```

6971 \newenvironment{sproblem}[1][{}]{
6972   \__problems_prob_args:n{#1}%\sref@target%
6973   \@in@omtexttrue% we are in a statement (for inline definitions)
6974   \stepcounter{problem}\record@problem
6975   \def\current@section@level{\prob@problem@kw}
6976   \tl_if_exist:NTF \l__problems_inclprob_type_tl {
6977     \tl_set_eq:NN \sproblemtype \l__problems_inclprob_type_tl
6978   }{
6979     \tl_set_eq:NN \sproblemtype \l__problems_prob_type_tl
6980   }
6981   \str_if_exist:NTF \l__problems_inclprob_id_str {
6982     \str_set_eq:NN \sproblemid \l__problems_inclprob_id_str
6983   }{
6984     \str_set_eq:NN \sproblemid \l__problems_prob_id_str
6985   }
6986
6987
6988   \clist_set:No \l_tmpa_clist \sproblemtype
6989   \tl_clear:N \l_tmpa_tl
6990   \clist_map_inline:Nn \l_tmpa_clist {
6991     \tl_if_exist:cT {\__problems_sproblem_##1_start:}{
6992       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_start:}}
6993     }
6994   }
6995   \tl_if_empty:NTF \l_tmpa_tl {
6996     \__problems_sproblem_start:
6997   }{
6998     \l_tmpa_tl
6999   }
7000   \stex_ref_new_doc_target:n \sproblemid
7001 }{
7002   \clist_set:No \l_tmpa_clist \sproblemtype
7003   \tl_clear:N \l_tmpa_tl
7004   \clist_map_inline:Nn \l_tmpa_clist {
7005     \tl_if_exist:cT {\__problems_sproblem_##1_end:}{
7006       \tl_set:Nn \l_tmpa_tl {\use:c{\__problems_sproblem_##1_end:}}
7007     }

```



```

7008 }
7009 \tl_if_empty:NTF \l_tmpa_tl {
7010   \__problems_sproblem_end:
7011 }{
7012   \l_tmpa_tl
7013 }
7014
7015
7016 \smallskip
7017 }
7018
7019
7020 \cs_new_protected:Nn \__problems_sproblem_start: {
7021   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
7022 }
7023 \cs_new_protected:Nn \__problems_sproblem_end: {\par\smallskip}
7024
7025 \newcommand\stexpatchproblem[3][] {
7026   \str_set:Nx \l_tmpa_str{ #1 }
7027   \str_if_empty:NTF \l_tmpa_str {
7028     \tl_set:Nn \__problems_sproblem_start: { #2 }
7029     \tl_set:Nn \__problems_sproblem_end: { #3 }
7030   }{
7031     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_start:\endcsname{ #2 }
7032     \exp_after:wN \tl_set:Nn \csname __problems_sproblem_#1_end:\endcsname{ #3 }
7033   }
7034 }
7035
7036
7037 \bool_if:NT \c__problems_boxed_bool {
7038   \surroundwithmdframed{problem}
7039 }

```

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

```

7040 \def\record@problem{
7041   \protected@write\@auxout{}
7042   {
7043     \string\@problem{\prob@number}
7044     {
7045       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7046         \l__problems_inclprob_pts_tl
7047       }{
7048         \l__problems_prob_pts_tl
7049       }
7050     }%
7051     {
7052       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7053         \l__problems_inclprob_min_tl
7054       }{
7055         \l__problems_prob_min_tl
7056       }
7057     }
7058   }
7059 }

```

(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).

```
7060 \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.

```
7061 \keys_define:nn { problem / solution }{
7062   id                .str_set:N = \l__problems_solution_id_str ,
7063   for               .tl_set:N  = \l__problems_solution_for_tl ,
7064   height            .dim_set:N = \l__problems_solution_height_dim ,
7065   creators          .clist_set:N = \l__problems_solution_creators_clist ,
7066   contributors      .clist_set:N = \l__problems_solution_contributors_clist ,
7067   srccite           .tl_set:N   = \l__problems_solution_srccite_tl
7068 }
7069 \cs_new_protected:Nn \__problems_solution_args:n {
7070   \str_clear:N \l__problems_solution_id_str
7071   \tl_clear:N \l__problems_solution_for_tl
7072   \tl_clear:N \l__problems_solution_srccite_tl
7073   \clist_clear:N \l__problems_solution_creators_clist
7074   \clist_clear:N \l__problems_solution_contributors_clist
7075   \dim_zero:N \l__problems_solution_height_dim
7076   \keys_set:nn { problem / solution }{ #1 }
7077 }
```

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

```
7078 \newcommand\@startsolution[1][{}]{
7079   \__problems_solution_args:n { #1 }
7080   \@in@omtexttrue% we are in a statement.
7081   \bool_if:NF \c__problems_boxed_bool { \hrule }
7082   \smallskip\noindent
7083   {\textbf\prob@solution@kw : \enspace}
7084   \begin{small}
7085   \def\current@section@level{\prob@solution@kw}
7086   \ignorespacesandpars
7087 }
```

\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.

```
7088 \newcommand\startsolutions{
7089   \specialcomment{solution}{\@startsolution}{
7090     \bool_if:NF \c__problems_boxed_bool {
7091       \hrule\medskip
7092     }
7093     \end{small}%
7094   }
7095   \bool_if:NT \c__problems_boxed_bool {
7096     \surroundwithmdframed{solution}
7097   }
7098 }
```

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

\stopsolutions

```
7099 \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.

```
7100 \ifsolutions
7101   \startsolutions
7102 \else
7103   \stopsolutions
7104 \fi
```

exnote

```
7105 \bool_if:NTF \c__problems_notes_bool {
7106   \newenvironment{exnote}[1][]{
7107     \par\smallskip\hrule\smallskip
7108     \noindent\textbf{\prob@note@kw : }\small
7109   }{
7110     \smallskip\hrule
7111   }
7112 }{
7113   \excludecomment{exnote}
7114 }
```

hint

```
7115 \bool_if:NTF \c__problems_notes_bool {
7116   \newenvironment{hint}[1][]{
7117     \par\smallskip\hrule\smallskip
7118     \noindent\textbf{\prob@hint@kw :~ }\small
7119   }{
7120     \smallskip\hrule
7121   }
7122   \newenvironment{exhint}[1][]{
7123     \par\smallskip\hrule\smallskip
7124     \noindent\textbf{\prob@hint@kw :~ }\small
7125   }{
7126     \smallskip\hrule
7127   }
7128 }{
7129   \excludecomment{hint}
7130   \excludecomment{exhint}
7131 }
```

gnote

```
7132 \bool_if:NTF \c__problems_notes_bool {
7133   \newenvironment{gnote}[1][]{
7134     \par\smallskip\hrule\smallskip
7135     \noindent\textbf{\prob@gnote@kw : }\small
7136   }{
7137     \smallskip\hrule
7138   }
7139 }{
7140   \excludecomment{gnote}
7141 }
```

39.3 Multiple Choice Blocks

EdN:17

mcb 17

```

7142 \newenvironment{mcb}{
7143   \begin{enumerate}
7144 }{
7145   \end{enumerate}
7146 }
```

we define the keys for the mcc macro

```

7147 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
7148   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
7149     \bool_set_true:N #1
7150   }{
7151     \bool_set_false:N #1
7152   }
7153 }
7154 \keys_define:nn { problem / mcc }{
7155   id          .str_set_x:N = \l__problems_mcc_id_str ,
7156   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
7157   T           .default:n    = { true } ,
7158   T           .bool_set:N   = \l__problems_mcc_t_bool ,
7159   F           .default:n    = { true } ,
7160   F           .bool_set:N   = \l__problems_mcc_f_bool ,
7161   Ttext       .code:n       = {
7162     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
7163   } ,
7164   Ftext       .code:n       = {
7165     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
7166   }
7167 }
7168 \cs_new_protected:Nn \l__problems_mcc_args:n {
7169   \str_clear:N \l__problems_mcc_id_str
7170   \tl_clear:N \l__problems_mcc_feedback_tl
7171   \bool_set_true:N \l__problems_mcc_t_bool
7172   \bool_set_true:N \l__problems_mcc_f_bool
7173   \bool_set_true:N \l__problems_mcc_Ttext_bool
7174   \bool_set_false:N \l__problems_mcc_Ftext_bool
7175   \keys_set:nn { problem / mcc }{ #1 }
7176 }
```

\mcc

```

7177 \newcommand\mcc[2][] {
7178   \l__problems_mcc_args:n{ #1 }
7179   \item #2
7180   \ifsolutions
7181     \\\
7182     \bool_if:NT \l__problems_mcc_t_bool {
7183       % TODO!
7184       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
7185     }
7186     \bool_if:NT \l__problems_mcc_f_bool {
```

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

```

7187      % TODO!
7188      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
7189    }
7190    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
7191      !
7192    }{
7193      \l__problems_mcc_feedback_tl
7194    }
7195    \fi
7196  } %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.

```

7197
7198 \keys_define:nn{ problem / inclproblem }{
7199   id      .str_set:N = \l__problems_inclprob_id_str,
7200   pts     .tl_set:N  = \l__problems_inclprob_pts_tl,
7201   min     .tl_set:N  = \l__problems_inclprob_min_tl,
7202   title   .tl_set:N  = \l__problems_inclprob_title_tl,
7203   refnum  .int_set:N  = \l__problems_inclprob_refnum_int,
7204   type    .tl_set:N  = \l__problems_inclprob_type_tl,
7205   mhrepos .str_set:N  = \l__problems_inclprob_mhrepos_str
7206 }
7207 \cs_new_protected:Nn \l__problems_inclprob_args:n {
7208   \str_clear:N \l__problems_prob_id_str
7209   \tl_clear:N \l__problems_inclprob_pts_tl
7210   \tl_clear:N \l__problems_inclprob_min_tl
7211   \tl_clear:N \l__problems_inclprob_title_tl
7212   \tl_clear:N \l__problems_inclprob_type_tl
7213   \int_zero_new:N \l__problems_inclprob_refnum_int
7214   \str_clear:N \l__problems_inclprob_mhrepos_str
7215   \keys_set:nn { problem / inclproblem }{ #1 }
7216   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
7217     \let\l__problems_inclprob_pts_tl\undefined
7218   }
7219   \tl_if_empty:NT \l__problems_inclprob_min_tl {
7220     \let\l__problems_inclprob_min_tl\undefined
7221   }
7222   \tl_if_empty:NT \l__problems_inclprob_title_tl {
7223     \let\l__problems_inclprob_title_tl\undefined
7224   }
7225   \tl_if_empty:NT \l__problems_inclprob_type_tl {
7226     \let\l__problems_inclprob_type_tl\undefined
7227   }
7228   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
7229     \let\l__problems_inclprob_refnum_int\undefined
7230   }
7231 }

```

```

7232
7233 \cs_new_protected:Nn \__problems_inclprob_clear: {
7234   \let\l__problems_inclprob_id_str\undefined
7235   \let\l__problems_inclprob_pts_tl\undefined
7236   \let\l__problems_inclprob_min_tl\undefined
7237   \let\l__problems_inclprob_title_tl\undefined
7238   \let\l__problems_inclprob_type_tl\undefined
7239   \let\l__problems_inclprob_refnum_int\undefined
7240   \let\l__problems_inclprob_mhrepos_str\undefined
7241 }
7242 \__problems_inclprob_clear:
7243
7244 \newcommand\includeproblem[2][ ]{
7245   \__problems_inclprob_args:n{ #1 }
7246   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
7247     \input{#2}
7248   }{
7249     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
7250       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
7251     }
7252   }
7253   \__problems_inclprob_clear:
7254 }

```

(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.

```

7255 \AddToHook{enddocument}{
7256   \bool_if:NT \c__problems_pts_bool {
7257     \message{Total:~\arabic{pts}~points}
7258   }
7259   \bool_if:NT \c__problems_min_bool {
7260     \message{Total:~\arabic{min}~minutes}
7261   }
7262 }

```

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

```

7263 \def\pts#1{
7264   \bool_if:NT \c__problems_pts_bool {
7265     \marginpar{#1~\prob@pt@kw}
7266   }
7267 }
7268 \def\min#1{
7269   \bool_if:NT \c__problems_min_bool {
7270     \marginpar{#1~\prob@min@kw}
7271   }
7272 }

```

`\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.

```

7273 \newcounter{pts}
7274 \def\show@pts{
7275   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
7276     \bool_if:NT \c__problems_pts_bool {
7277       \marginpar{\l__problems_inclprob_pts_tl\ \prob@pt@kw\smallskip}
7278       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
7279     }
7280   }{
7281     \tl_if_exist:NT \l__problems_prob_pts_tl {
7282       \bool_if:NT \c__problems_pts_bool {
7283         \marginpar{\l__problems_prob_pts_tl\ \prob@pt@kw\smallskip}
7284         \addtocounter{pts}{\l__problems_prob_pts_tl}
7285       }
7286     }
7287   }
7288 }

```

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

`\show@min`

```

7289 \newcounter{min}
7290 \def\show@min{
7291   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
7292     \bool_if:NT \c__problems_min_bool {
7293       \marginpar{\l__problems_inclprob_min_tl\ min}
7294       \addtocounter{min}{\l__problems_inclprob_min_tl}
7295     }
7296   }{
7297     \tl_if_exist:NT \l__problems_prob_min_tl {
7298       \bool_if:NT \c__problems_min_bool {
7299         \marginpar{\l__problems_prob_min_tl\ min}
7300         \addtocounter{min}{\l__problems_prob_min_tl}
7301       }
7302     }
7303   }
7304 }
7305 </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.

```
7306 \@@=hwexam>
7307 \*cls>
7308 \ProvidesExplClass{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7309 \RequirePackage{l3keys2e}
7310 \DeclareOption*{
7311   \PassOptionsToClass{\CurrentOption}{document-structure}
7312   \PassOptionsToPackage{\CurrentOption}{stex}
7313   \PassOptionsToPackage{\CurrentOption}{hwexam}
7314   \PassOptionsToPackage{\CurrentOption}{tikzinput}
7315 }
7316 \ProcessOptions
```

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

```
7317 \LoadClass{document-structure}
7318 \RequirePackage{stex}
7319 \RequirePackage{hwexam}
7320 \RequirePackage{tikzinput}
7321 \RequirePackage{graphicx}
7322 \RequirePackage{a4wide}
7323 \RequirePackage{amssymb}
7324 \RequirePackage{amstext}
7325 \RequirePackage{amsmath}
```

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


```

7326 \newcommand\assig@default@type{\hwexam@assignment@kw}
7327 \def\document@hwexamtype{\assig@default@type}
7328 <@@=document_structure>
7329 \keys_define:nn { document-structure / document }{
7330 id .str_set_x:N = \c_document_structure_document_id_str,
7331 hwexamtype .tl_set:N = \document@hwexamtype
7332 }
7333 <@@=hwexam>
7334 </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.

```
7335 \*package>
7336 \ProvidesExplPackage{hwexam}{2022/02/26}{3.0.1}{homework assignments and exams}
7337 \RequirePackage{13keys2e}
7338
7339 \newif\iftest\testfalse
7340 \DeclareOption{test}{\testtrue}
7341 \newif\ifmultiple\multiplefalse
7342 \DeclareOption{multiple}{\multipletrue}
7343 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
7344 \ProcessOptions
```

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

```
7345 \RequirePackage{keyval}[1997/11/10]
7346 \RequirePackage{problem}
```

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

```
7347 \newcommand\hwexam@assignment@kw{Assignment}
7348 \newcommand\hwexam@given@kw{Given}
7349 \newcommand\hwexam@due@kw{Due}
7350 \newcommand\hwexam@testemptypage@kw{This~page~was~intentionally~left~
7351 blank~for~extra~space}
7352 \def\hwexam@minutes@kw{minutes}
7353 \newcommand\correction@probs@kw{prob.}
7354 \newcommand\correction@pts@kw{total}
7355 \newcommand\correction@reached@kw{reached}
7356 \newcommand\correction@sum@kw{Sum}
7357 \newcommand\correction@grade@kw{grade}
7358 \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

```

7359 \AddToHook{begindocument}{
7360 \ltx@ifpackageloaded{babel}{
7361 \makeatletter
7362 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
7363 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
7364 \input{hwexam-ngerman.ldf}
7365 }
7366 \clist_if_in:NnT \l_tmpa_clist {finnish}{
7367 \input{hwexam-finnish.ldf}
7368 }
7369 \clist_if_in:NnT \l_tmpa_clist {french}{
7370 \input{hwexam-french.ldf}
7371 }
7372 \clist_if_in:NnT \l_tmpa_clist {russian}{
7373 \input{hwexam-russian.ldf}
7374 }
7375 \makeatother
7376 }{}
7377 }
7378

```

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.

```

7379 \newcounter{assignment}
7380 \numberproblemsin{assignment}
7381 \renewcommand\prob@label[1]{\assignment@number.#1}

```

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

```

7382 \keys_define:nn { hwexam / assignment } {
7383 id .str_set:N = \l__hwexam_assign_id_str,
7384 number .int_set:N = \l__hwexam_assign_number_int,
7385 title .tl_set:N = \l__hwexam_assign_title_tl,
7386 type .tl_set:N = \l__hwexam_assign_type_tl,
7387 given .tl_set:N = \l__hwexam_assign_given_tl,
7388 due .tl_set:N = \l__hwexam_assign_due_tl,
7389 loadmodules .code:n = {
7390 \bool_set_true:N \l__hwexam_assign_loadmodules_bool
7391 }
7392 }
7393 \cs_new_protected:Nn \__hwexam_assignment_args:n {
7394 \str_clear:N \l__hwexam_assign_id_str
7395 \int_set:Nn \l__hwexam_assign_number_int {-1}
7396 \tl_clear:N \l__hwexam_assign_title_tl
7397 \tl_clear:N \l__hwexam_assign_type_tl
7398 \tl_clear:N \l__hwexam_assign_given_tl
7399 \tl_clear:N \l__hwexam_assign_due_tl
7400 \bool_set_false:N \l__hwexam_assign_loadmodules_bool

```

```

7401 \keys_set:nn { hwexam / assignment }{ #1 }
7402 }

```

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.

```

7403 \newcommand\given@due[2]{
7404 \bool_lazy_all:nF {
7405 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7406 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7407 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7408 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7409 }{ #1 }
7410
7411 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
7412 \tl_if_empty:NF \l__hwexam_assign_given_tl {
7413 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
7414 }
7415 }{
7416 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
7417 }
7418
7419 \bool_lazy_or:nnF {
7420 \bool_lazy_and_p:nn {
7421 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7422 }{
7423 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7424 }
7425 }{
7426 \bool_lazy_and_p:nn {
7427 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
7428 }{
7429 \tl_if_empty_p:V \l__hwexam_assign_due_tl
7430 }
7431 }{ ,~ }
7432
7433 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
7434 \tl_if_empty:NF \l__hwexam_assign_due_tl {
7435 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
7436 }
7437 }{
7438 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
7439 }
7440
7441 \bool_lazy_all:nF {
7442 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
7443 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
7444 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
7445 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
7446 }{ #2 }
7447 }

```

`\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.

```

7448 \newcommand\assignment@title[3]{
7449 \tl_if_empty:NTF \l__hwexam_inclasssign_title_tl {
7450 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
7451 #1
7452 }{
7453 #2\l__hwexam_assign_title_tl#3
7454 }
7455 }{
7456 #2\l__hwexam_inclasssign_title_tl#3
7457 }
7458 }

```

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

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

```

7459 \newcommand\assignment@number{
7460 \int_compare:nNnTF \l__hwexam_inclasssign_number_int = {-1} {
7461 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7462 \arabic{assignment}
7463 } {
7464 \int_use:N \l__hwexam_assign_number_int
7465 }
7466 }{
7467 \int_use:N \l__hwexam_inclasssign_number_int
7468 }
7469 }

```

(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.

```

7470 \newenvironment{assignment}[1][ ]{
7471 \__hwexam_assignment_args:n { #1 }
7472 %\sref@target
7473 \int_compare:nNnTF \l__hwexam_assign_number_int = {-1} {
7474 \global\stepcounter{assignment}
7475 }{
7476 \global\setcounter{assignment}{\int_use:N\l__hwexam_assign_number_int}
7477 }
7478 \setcounter{problem}{0}
7479 \def\current@section@level{\document@hwexamtype}
7480 %\sref@label@id{\document@hwexamtype \thesection}
7481 \begin{@assignment}
7482 }{
7483 \end{@assignment}
7484 }

```

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

```

7485 \def\ass@title{
7486 \protect\document@hwexamtype~\arabic{assignment}
7487 \assignment@title{}\{;\}{} -- \given@due{}\}{}
7488 }
7489 \ifmultiple
7490 \newenvironment{@assignment}{
7491 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
7492 \begin{sfragment}[loadmodules]{\ass@title}
7493 }{
7494 \begin{sfragment}{\ass@title}
7495 }
7496 }{
7497 \end{sfragment}
7498 }

```

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

```

7499 \else
7500 \newenvironment{@assignment}{
7501 \begin{center}\bf
7502 \Large@title\strut\
7503 \document@hwexamtype~\arabic{assignment}\assignment@title{}\{;\}{}{\}{}
7504 \large\given@due{--;\}{}\}{}
7505 \end{center}
7506 }{}
7507 \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.

```

7508 \keys_define:nn { hwexam / inclassignment } {
7509 %id .str_set_x:N = \l__hwexam_assign_id_str,
7510 number .int_set:N = \l__hwexam_inclassign_number_int,
7511 title .tl_set:N = \l__hwexam_inclassign_title_tl,
7512 type .tl_set:N = \l__hwexam_inclassign_type_tl,
7513 given .tl_set:N = \l__hwexam_inclassign_given_tl,
7514 due .tl_set:N = \l__hwexam_inclassign_due_tl,
7515 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
7516 }
7517 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
7518 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
7519 \tl_clear:N \l__hwexam_inclassign_title_tl
7520 \tl_clear:N \l__hwexam_inclassign_type_tl
7521 \tl_clear:N \l__hwexam_inclassign_given_tl
7522 \tl_clear:N \l__hwexam_inclassign_due_tl
7523 \str_clear:N \l__hwexam_inclassign_mhrepos_str
7524 \keys_set:nn { hwexam / inclassignment }{ #1 }
7525 }
7526 \__hwexam_inclassignment_args:n {}
7527
7528 \newcommand\inputassignment[2][ ]{

```

```

7529 \_hwexam_inclassnment_args:n { #1 }
7530 \str_if_empty:NTF \l__hwexam_inclassn_mhrepos_str {
7531 \input{#2}
7532 }{
7533 \stex_in_repository:nn{\l__hwexam_inclassn_mhrepos_str}{
7534 \input{\mhp{path}\l__hwexam_inclassn_mhrepos_str}{#2}}
7535 }
7536 }
7537 \_hwexam_inclassnment_args:n {}
7538 }
7539 \newcommand\includeassignment[2][]{
7540 \newpage
7541 \inputassignment[#1]{#2}
7542 }

```

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

41.4 Typesetting Exams

\quizheading

```

7543 \ExplSyntaxOff
7544 \newcommand\quizheading[1]{%
7545 \def\@tas{#1}%
7546 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
7547 \ifx\@tas\@empty\else%
7548 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
7549 \fi%
7550 }
7551 \ExplSyntaxOn

```

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

\testheading

```

7552
7553 \def\hwexamheader{\input{hwexam-default.header}}
7554
7555 \def\hwexamminutes{
7556 \tl_if_empty:NTF \testheading@duration {
7557 {\testheading@min}~\hwexam@minutes@kw
7558 }{
7559 \testheading@duration
7560 }
7561 }
7562
7563 \keys_define:nn { hwexam / testheading } {
7564 min .tl_set:N = \testheading@min,
7565 duration .tl_set:N = \testheading@duration,
7566 reqpts .tl_set:N = \testheading@reqpts,
7567 tools .tl_set:N = \testheading@tools
7568 }
7569 \cs_new_protected:Nn \_hwexam_testheading_args:n {
7570 \tl_clear:N \testheading@min
7571 \tl_clear:N \testheading@duration

```

```

7572 \tl_clear:N \testheading@reqpts
7573 \tl_clear:N \testheading@tools
7574 \keys_set:nn { hwexam / testheading }{ #1 }
7575 }
7576 \newenvironment{testheading}[1][]{
7577   \_hwexam_testheading_args:n{ #1 }
7578   \newcount\check@time\check@time=\testheading@min
7579   \advance\check@time by -\theassignment@totalmin
7580   \newif\if@bonuspoints
7581   \tl_if_empty:NTF \testheading@reqpts {
7582     \@bonuspointsfalse
7583   }{
7584     \newcount\bonus@pts
7585     \bonus@pts=\theassignment@totalpts
7586     \advance\bonus@pts by -\testheading@reqpts
7587     \edef\bonus@pts{\the\bonus@pts}
7588     \@bonuspointstrue
7589   }
7590   \edef\check@time{\the\check@time}
7591
7592   \makeatletter\hwexamheader\makeatother
7593 }{
7594   \newpage
7595 }

```

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

\testspace

```

7596 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

7597 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

7598 \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.

```

7599 <@=problems>
7600 \renewcommand\@problem[3]{
7601   \stepcounter{assignment@probs}
7602   \def\__problemspts{#2}
7603   \ifx\__problemspts\@empty\else
7604     \addtocounter{assignment@totalpts}{#2}
7605   \fi
7606   \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
7607   \xdef\correction@probs{\correction@probs & #1}%
7608   \xdef\correction@pts{\correction@pts & #2}
7609   \xdef\correction@reached{\correction@reached &}

```



```

7610 }
7611 <@@=hwexam>

```

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

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

```

7612 \newcounter{assignment@probs}
7613 \newcounter{assignment@totalpts}
7614 \newcounter{assignment@totalmin}
7615 \def\correction@probs{\correction@probs@kw}
7616 \def\correction@pts{\correction@pts@kw}
7617 \def\correction@reached{\correction@reached@kw}
7618 \stepcounter{assignment@probs}
7619 \newcommand\correction@table{
7620 \resizebox{\textwidth}{!}{%
7621 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
7622 &\multicolumn{\theassignment@probs}{c|}||%|
7623 {\footnotesize\correction@forgrading@kw} &\\ \hline
7624 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
7625 \correction@pts & \theassignment@totalpts & \\ \hline
7626 \correction@reached & & \[.7cm]\hline
7627 \end{tabular}}
7628 </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}

```