

The \TeX 3 Package *

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

2022-01-13

Abstract

TODO

*Version 3.0 (last revised 2022-01-13)

Contents

I	Manual	1
1	Stuff	2
1.1	Modules	2
1.1.1	Semantic Macros and Notations	2
	Other Argument Types	4
	Precedences	6
1.1.2	Archives and Imports	6
	Namespaces	6
	Paths in Import-Statements	7
II	Documentation	8
2	sTeX-Basics	9
2.1	Macros and Environments	9
3	sTeX-MathHub	11
3.1	Macros and Environments	11
3.1.1	Files, Paths, URIs	11
3.1.2	MathHub Archives	12
4	sTeX-References	14
4.1	Macros and Environments	14
5	sTeX-Modules	15
5.1	Macros and Environments	15
5.1.1	The module-environment	17
6	sTeX-Module Inheritance	20
6.1	Macros and Environments	20
6.1.1	SMS Mode	20
6.1.2	Imports and Inheritance	21
7	sTeX-Symbols	24
7.1	Macros and Environments	24
8	sTeX-Terms	27
8.1	Macros and Environments	27
9	sTeX-Structural Features	30
9.1	Macros and Environments	30
9.1.1	Structures	30
10	sTeX-Statements	31
10.1	Macros and Environments	31

11	STeX-Proofs: Structural Markup for Proofs	32
11.1	Introduction	34
11.2	The User Interface	35
11.2.1	Package Options	35
11.2.2	Proofs and Proof steps	35
11.2.3	Justifications	35
11.2.4	Proof Structure	36
11.2.5	Proof End Markers	37
11.2.6	Configuration of the Presentation	37
11.3	Limitations	37
12	STeX-Metatheory	39
12.1	Symbols	39
III	Extensions	40
13	Tikzinput	41
13.1	Macros and Environments	41
14	document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX	42
14.1	Introduction	42
14.2	The User Interface	43
14.2.1	Package and Class Options	43
14.2.2	Document Structure	43
14.2.3	Ignoring Inputs	44
14.2.4	Structure Sharing	45
14.2.5	Global Variables	45
14.2.6	Colors	46
14.3	Limitations	46
15	Slides and Course Notes	47
15.1	Introduction	47
15.2	The User Interface	47
15.2.1	Package Options	47
15.2.2	Notes and Slides	48
15.2.3	Header and Footer Lines of the Slides	49
15.2.4	Frame Images	49
15.2.5	Colors and Highlighting	50
15.2.6	Front Matter, Titles, etc.	50
15.2.7	Excursions	50
15.2.8	Miscellaneous	50
15.3	Limitations	50

16	problem.sty: An Infrastructure for formatting Problems	51
16.1	Introduction	51
16.2	The User Interface	51
16.2.1	Package Options	51
16.2.2	Problems and Solutions	52
16.2.3	Multiple Choice Blocks	53
16.2.4	Including Problems	53
16.2.5	Reporting Metadata	53
16.3	Limitations	53
17	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	55
17.1	Introduction	56
17.2	The User Interface	56
17.2.1	Package and Class Options	56
17.2.2	Assignments	56
17.2.3	Typesetting Exams	56
17.2.4	Including Assignments	57
17.3	Limitations	57
IV	Implementation	59
18	gTeX-Basics Implementation	60
18.1	The gTeXDocument Class	60
18.2	Preliminaries	60
18.3	Messages and logging	61
18.4	Persistence	62
18.5	HTML Annotations	62
18.6	Languages	65
18.7	Activating/Deactivating Macros	66
19	gTeX-MathHub Implementation	68
19.1	Generic Path Handling	68
19.2	PWD and kpsewhich	70
19.3	File Hooks and Tracking	71
19.4	MathHub Repositories	72
20	gTeX-References Implementation	78
20.1	Document URIs and URLs	78
20.2	Setting Reference Targets	80
20.3	Using References	81
21	gTeX-Modules Implementation	83
21.1	The module environment	86
21.2	Invoking modules	91
22	gTeX-Module Inheritance Implementation	93
22.1	SMS Mode	93
22.2	Inheritance	97

23	STEX-Symbols Implementation	102
23.1	Symbol Declarations	102
23.2	Notations	108
24	STEX-Terms Implementation	116
24.1	Symbol Invocations	116
24.2	Terms	119
24.3	Notation Components	125
25	STEX-Structural Features Implementation	128
25.1	The feature environment	128
25.2	Features	130
26	STEX-Statements Implementation	135
26.1	Definitions	136
26.2	Assertions	138
26.3	Examples	140
26.4	OMText	141
27	The Implementation	142
27.1	Package Options	142
27.2	Proofs	142
27.3	Justifications	148
28	STEX-Others Implementation	150
29	STEX-Metatheory Implementation	151
30	Tikzinput Implementation	154
31	document-structure.sty Implementation	156
31.1	The OMDoc Class	156
31.2	Class Options	156
31.3	Beefing up the document environment	157
31.4	Implementation: OMDoc Package	157
31.5	Package Options	157
31.6	Document Structure	159
31.7	Front and Backmatter	162
31.8	Global Variables	164
32	MiKoSlides – Implementation	165
32.1	Class and Package Options	165
32.2	Notes and Slides	167
32.3	Header and Footer Lines	171
32.4	Frame Images	172
32.5	Colors and Highlighting	173
32.6	Sectioning	174
32.7	Excursions	176

33 The Implementation	178
33.1 Package Options	178
33.2 Problems and Solutions	179
33.3 Multiple Choice Blocks	184
33.4 Including Problems	185
33.5 Reporting Metadata	186
34 Implementation: The hwexam Class	188
34.1 Class Options	188
35 Implementation: The hwexam Package	190
35.1 Package Options	190
35.2 Assignments	191
35.3 Including Assignments	194
35.4 Typesetting Exams	195
35.5 Leftovers	197

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

To declare a symbol (in a module), we use `\symdecl`, which takes as argument the name of the corresponding semantic macro, e.g. `\symdecl{foo}` introduces the macro `\foo`. Additionally, `\symdecl` takes several options, the most important one being its arity. `foo` as declared above yields a *constant* symbol. To introduce an *operator* which takes arguments, we have to specify which arguments it takes.

For example, to introduce binary multiplication, we can do `\symdecl[args=2]{mult}`. We can then supply the semantic macro with arbitrarily many notations, such as `\notation{mult}{#1 #2}`.

Example 1

```
\symdecl[args=2]{mult}
\notation{mult}{#1 #2}
 $\mult{a}{b}$ 
```

ab

Since usually, a freshly introduced symbol also comes with a notation from the start, the `\symdef` command combines `\symdecl` and `\notation`. So instead of the above, we could have also written

```
\symdef[args=2]{mult}{#1 #2}
```


Adding more notations like `\notation[cdot]{mult}{#1 \comp{\cdot} #2}` or `\notation[times]{mult}{#1 \comp{\times} #2}` allows us to write $\mult[cdot]{a}{b}$ and $\mult[times]{a}{b}$:

Example 2

```
\notation[cdot]{mult}{#1 \comp{\cdot} #2}
\notation[times]{mult}{#1 \comp{\times} #2}
 $\mult[cdot]{a}{b}$  and  $\mult[times]{a}{b}$ 
```

$a \cdot b$ and $a \times b$

.

Not using an explicit option with a semantic macro yields the first declared notation, unless changed¹.

Outside of math mode, or by using the starred variant `\foo*`, allows to provide a custom notation, where notational (or textual) components can be given explicitly in square brackets.

Example 3

```
 $\mult*{a}[\comp{\ast}]{b}$  is the
\mult[\comp{product of}][ $\$a$ ][\comp{and}][ $\$b$ ]
```

$a * b$ is the product of a and b

.

In custom mode, prefixing an argument with a star will not print that argument, but still export it to OMDoc:

Example 4

```
\mult[\comp{Multiplying}]* $\mult{a}{b}$ [ again by  $\$b$  ] yields ...
```

Multiplying again by b yields...

The syntax `*[int]` allows switching the order of arguments. For example, given a 2-ary semantic macro `\forevery` with exemplary notation `\forall #1. #2`, we can write

Example 5

```
\symdecl[ args=2]{forevery}
\forevery* [2]{The proposition  $\$P$  }[\comp{holds for every} ]*[1]{ $\$x$  in  $A$  }
```

The proposition P holds for every $x \in A$

¹EdNOTE: TODO

When using `*[n]`, after reading the provided (n th) argument, the “argument counter” automatically continues where we left off, so the `*[1]` in the above example can be omitted.

For a macro with `arity > 0`, we can refer to the operator *itself* semantically by suffixing the semantic macro with an exclamation point `!` in either text or math mode. For that reason `\notation` (and thus `\symdef`) take an additional optional argument `op=`, which allows to assign a notation for the operator itself. e.g.

Example 6

```
\symdef[ args=2,op={+}]{add}{#1 \comp+ #2}
The operator  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{\text{add}}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

The operator $+$ adds two elements, as in $a+b$.

`*` is composable with `!` for custom notations, as in:

Example 7

```
\mult![\comp{Multiplication}] (denoted by  $\textcolor{teal}{\$}\textcolor{teal}{\text{mult}}\textcolor{teal}{*}\textcolor{teal}{![\comp{\cdots}]\textcolor{teal}{\$}}$  is defined by...
```

$\textcolor{teal}{\text{Multiplication}}$ (denoted by \cdot) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There’s also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}![some text]`

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters

representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments, \TeX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDoc and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 8

```
\symdef[ args=a]{mult}{#1}{#1 \comp\cdot #2}
$\mult{a,b,c,{d^e},f}$
```

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 9

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$\numseq{a,b,c}{\mathbb{R}}$
```

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

2 3

²EDNOTE: what about e.g. `\int _x \int _y \int _z f dx dy dz`?

³EDNOTE: “decompose” a-type arguments into fixed-arity operators?

Precedences

Every notation has an (upwards) *operator precedence* and for each argument a (downwards) *argument precedence* used for automated bracketing. For example, a notation for a binary operator `\foo` could be declared like this:

```
\notation[prec=200;500x600]{foo}{#1 \comp{+} #2}
```

assigning an operator precedence of 200, an argument precedence of 500 for the first argument, and an argument precedence of 600 for the second argument.

\TeX insert brackets thusly: Upon encountering a semantic macro (such as `\foo`), its operator precedence (e.g. 200) is compared to the current downwards precedence (initially `\neginfprec`). If the operator precedence is *larger* than the current downwards precedence, parentheses are inserted around the semantic macro.

Notations for symbols of arity 0 have a default precedence of `\infprec`, i.e. by default, parentheses are never inserted around constants. Notations for symbols with arity > 0 have a default operator precedence of 0. If no argument precedences are explicitly provided, then by default they are equal to the operator precedence.

Consequently, if some operator A should bind stronger than some operator B , then A as operator precedence should be smaller than B 's argument precedences.

For example:

Example 10

```
\notation[prec=100]{plus}{#1 \comp{+} #2}
\notation[prec=50]{times}{#1 \comp{\cdot} #2}
 $\plus{a}{\times{b}{c}}$  and  $\times{a}{\plus{b}{c}}$ 
```

$a+b \cdot c$ and $a \cdot (b+c)$

1.1.2 Archives and Imports

Namespaces

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{module}{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 suffix¹.

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.

Paths in Import-Statements

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

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

The module `Foo` must either be declared in the file `<top-directory>/some/path/Foo[.<lang>].tex`, or in `<top-directory>/some/path[.<lang>].tex` (which are checked in that order).

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

Since this is less compatible with a modular development, using full URIs directly is discouraged.

¹which is internally attached to the module name instead, but a user need not worry about that.

Part II
Documentation

Chapter 2

sTeX-Basics

Both the sTeX package and class offer the following package options:

debug ($\langle log-prefix \rangle *$) Logs debugging information with the given prefixes to the terminal, or all if **all** is given.

showmods ($\langle boolean \rangle$) Shows explicit module information at the document margins.

lang ($\langle language \rangle *$) Languages to load with the **babel** package.

mathhub ($\langle directory \rangle$) MathHub folder to search for repositories.

sms ($\langle boolean \rangle$) use *persisted* mode (see ???).

image ($\langle boolean \rangle$) passed on to tikzinput.

2.1 Macros and Environments

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

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

Logs $\langle message \rangle$, if the package option **debug** contains $\langle log-prefix \rangle$.

<code>\stex_add_to_sms:n</code>	Adds the provided code to the <code>.sms</code> -file of the document.
---------------------------------	--

<code>\if@latexml</code>	L ^A T _E X2e and L ^A T _E X3 conditionals for L ^A T _E XML.
<code>\latexml_if_p:</code>	
<code>\latexml_if:T</code>	
<code>\latexml_if:F</code>	
<code>\latexml_if:TF</code>	

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

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

Annotates the HTML generated by $\langle content \rangle$ with

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

`\stex_annotate_invisible:n` adds the attributes

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

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

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

<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`.

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

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

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

<code>\MSC</code>	<code>\MSC{⟨msc⟩}</code>
-------------------	--------------------------

Designates the *math subject classifier* of the current module / file.

Chapter 3

STEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

3.1 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.
--------------------------------	---

3.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$
<code>\stex_path_from_string:(NV cn cV)</code>	

turns the $\langle string \rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle path-variable \rangle$. Also applies `\stex_path_canonicalize:N`.

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	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</code>	\underline{TF} \star

Checks whether the path provided is *absolute*, i.e. starts with an empty segment

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

`\g_stex_currentfile_seq`

The file being currently processed (respecting `\input` etc.)

Test 1

```
\ExplSyntaxOn
\def\cpath@print#1{
\stex_path_from_string:Nn \l_tmpb_seq { #1 }
\stex_path_to_string:NN \l_tmpb_seq \l_tmpa_str
\str_use:N \l_tmpa_str
}
\ExplSyntaxOff
\begin{center}
\begin{tabular}{|l|l|l|}\hline
path & canonicalized path & expected\\\hline
aaa & \cpath@print{aaa} & aaa \\
.././aaa & \cpath@print{.././aaa} & & .././aaa \\
aaa/bbb & \cpath@print{aaa/bbb} & & aaa/bbb \\
aaa/. & \cpath@print{aaa/.} & & \\
.././aaa/bbb & \cpath@print{.././aaa/bbb} & & .././aaa/bbb \\
../aaa/./bbb & \cpath@print{../aaa/./bbb} & & ../bbb \\
../aaa/bbb & \cpath@print{../aaa/bbb} & & ../aaa/bbb \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/ddd \\
aaa/bbb/./ddd & \cpath@print{aaa/bbb/./ddd} & & aaa/bbb/ddd \\
./ & \cpath@print{./} & & \\
aaa/bbb/./.. & \cpath@print{aaa/bbb/./..} & & \\
\end{tabular}
\end{center}
```

path	canonicalized path	expected
aaa	aaa	aaa
.././aaa	.././aaa	.././aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/.		
.././aaa/bbb	.././aaa/bbb	.././aaa/bbb
../aaa/./bbb	../bbb	../bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb/./ddd	aaa/ddd	aaa/ddd
aaa/bbb/./ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb/./..		

3.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

We determine the path to the local MathHub folder via one of three means, in order of precedence:

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.

`\l_stex_current_repository_prop`

Always points to the *current* MathHub repository (if we currently are in one). Has the fields `id`, `ns` (namespace), `narr` (narrative namespace; currently not in use) and `deps` (dependencies; currently not in use).

<hr/> <hr/> <code>\stex_set_current_repository:n</code>	Sets the current repository to the one with the provided ID. calls <code>__stex_mathhub_do_manifest:n</code> , so works whether this repository's MANIFEST.MF-file has already been read or not.
<hr/> <hr/> <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 .sms-file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

\ExplSyntaxOn
\stex_require_repository:n { Foo/Bar }
id:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {id}\ \
narr:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {narr}\ \
ns:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {ns}\ \
deps:-\prop_item:cn {c_stex_mathhub_Foo/Bar_manifest_prop} {deps}\ \
\stex_require_repository:n { Bar/Foo }
\ExplSyntaxOff

```

```

id: Foo/Bar
narr:
ns: http://mathhub.info/tests/Foo/Bar
deps:

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

All information of a module is stored as a property list. `\l_stex_current_module_prop` always points to the current module (if existent).

Most importantly, the `content`-field stores all the code to execute on activation; i.e. when this module is being included.

Additionally, it stores:

- The *name* in field `name`,
- the *namespace* in field `ns`,
- this module's *language* in field `lang`,
- if a language module that translates some other modules, the *original* module in field `sig` (for signature),
- the *metatheory* in field `meta`,
- the URIs of all *imported modules* in field `imports`,
- the names of all *declarations* in field `constants`,
- the *file* this module was declared in in field `file`,

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

A property list mapping file paths to the lists of all modules declared therein. `\g_stex_modules_in_file_seq` always points to the current file(-stream - `\inputs` are considered the same file).

```
\stex_if_in_module_p: * Conditional for whether we are currently in a module
\stex_if_in_module:TF *
```

```
\stex_if_module_exists_p:n *
\stex_if_module_exists:nTF *
```

Conditional for whether a module with the provided URI is already known.

```
\stex_add_to_current_module:n
\STEXexport
```

Adds the provided tokens to the `content` field of the current module.

```
\stex_add_constant_to_current_module:n
```

Adds the declaration with the provided name to the `constants` field of the current module.

```
\stex_add_import_to_current_module:n
```

Adds the module with the provided full URI to the `imports` field of the current module.

```
\stex_modules_compute_namespace:nN \stex_modules_compute_namespace:nN
{\<namespace>} {\<path>}
```

Computes the namespace for file `<path>` in repository with namespace `<namespace>` as follows:

If the file is `.../source/sub/file.tex` and the namespace `http://some.namespace/foo`, then the namespace of is `http://some.namespace/foo/sub/file`.

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

```
\ExplSyntaxOn
\stex_modules_current_namespace:
Namespace~1:\\ \l_stex_modules_ns_str \\
Faking~a~repository:\\
\stex_set_current_repository:n{Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \testtemp
\edef\testtempb{\detokenize{source}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\edef\testtempb{\detokenize{test}}
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtempb }
\exp_args:NNo \seq_put_right:Nn \g_stex_currentfile_seq { \testtemp }
\stex_modules_current_namespace:
Namespace~2:\\ \l_stex_modules_ns_str
\ExplSyntaxOff
```

```

Namespace 1:
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest
Faking a repository:
Namespace 2:
http://mathhub.info/tests/Foo/Bar/test/stextest

```

.

5.1.1 The module-environment

`module` `\begin{module}[\langle options \rangle]{\langle name \rangle}`
 Opens a new module with name $\langle name \rangle$.
 TODO document options.

`\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_prop` appropriately.

`\stex_modules_heading:` Takes care of the module header, if the `showmods` package option is true. This macro can be overridden for customization.

`@module` `\begin{@module}[\langle options \rangle]{\langle name \rangle}`
 Core functionality of the `module-environment` without a header.

Test 4

```

\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\begin{@module}{Foo}
Module~path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{@module}
\ExplSyntaxOff

```

```

Module path: http://mathhub.info/tests/Foo/Bar?Foo
Language:
Signature:
Metatheory:

```

.

Test 5

```
\ExplSyntaxOn
\stex_set_current_repository:n {Foo/Bar}
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\seq_pop_right:NN \g_stex_currentfile_seq \l_tmpa_tl
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{tests} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Bar} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{source} }
\seq_put_right:Nx \g_stex_currentfile_seq { \tl_to_str:n{Foo.tex} }
\stex_debug:nn{modules}{Test:-\stex_path_to_string:N \g_stex_currentfile_seq }
\begin{module}[title=Foo Bar]{Bar}
Module-path:-
\prop_item:Nn \l_stex_current_module_prop { ns }?
\prop_item:Nn \l_stex_current_module_prop { name }\\
Language:-\prop_item:Nn \l_stex_current_module_prop { lang }\\
Signature:-\prop_item:Nn \l_stex_current_module_prop { sig }\\
Metatheory:-\prop_item:Nn \l_stex_current_module_prop { meta }\\
\end{module}
\ExplSyntaxOff
```

```
Module 5.1.1[Bar] (FooBar)
Module path: http://mathhub.info/tests/Foo/Bar/Foo?Bar
Language:
Signature:
Metatheory:
```

\STEXModule \STEXModule {*⟨fragment⟩*}

Attempts to find a module whose URI ends with *⟨fragment⟩* 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 `?{⟨symbolname⟩}`. In the first case, it stores the full URI in *⟨macro⟩*; in the second case, it invokes the symbol *⟨symbolname⟩* in the selected module.

Test 6

```
\begin{module}{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest2}
\importmodule{STEXModuleTest1}
\symdecl{foo}
\end{module}
\begin{module}{STEXModuleTest3}
\importmodule{STEXModuleTest2}
\symdecl{foo}
\STEXModule{STEXModuleTest1}!\teststring
\teststring\
\STEXModule{STEXModuleTest2}!\teststring
\teststring\
\STEXModule{STEXModuleTest3}!\teststring
\teststring\
\STEXModule{STEXModuleTest1}?{foo}[\comp{foo1}]\
\STEXModule{STEXModuleTest2}?{foo}[\comp{foo2}]\
\STEXModule{STEXModuleTest3}?{foo}[\comp{foo3}]\
\end{module}
```




`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 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:

`\g_stex_smsmode_allowedmacros_tl`

Macros that are executed as is; i.e. with the category code scheme used in SMS mode.

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

Importantly, these macros need to call `\stex_smsmode_set_codes:` after reading all arguments. Note, that `\stex_smsmode_set_codes:` takes care of checking whether we are in SMS mode in the first place, so calling this function eagerly is unproblematic.

`\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_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

Sets the current category code scheme to that of the SMS mode, if SMS mode is currently active and if necessary.

This method should be called at the end of every macro or `\begin` environment code that are allowed in SMS mode.

`\stex_in_smsmode:nn`

`\stex_in_smsmode:nn {<name>} {<code>}`

Executes `<code>` in SMS mode. `<name>` can be arbitrary, but should be distinct, since it allows for nesting `\stex_in_smsmode:nn` without spuriously terminating SMS mode.

Test 7

```
\immediate\openout\testfile=./tests/sometest.tex
\immediate\write\testfile{\detokenize{\this is \a test}^J}
\immediate\write\testfile{\detokenize{this \is a \test}}
\immediate\closeout\testfile
\ExplSyntaxOn
\stex_in_smsmode:nn { foo } {
\input{tests/sometest.tex}
}
\ExplSyntaxOff
```

6.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

```
\begin{module}{Foo}
\symdecl[name=foo, args=3]{bar}
\symdecl[ args=bai]{foobar}
Meaning:-\present\bar\
\end{module}
Meaning:-\present\bar\
\begin{module}{Importtest}
\importmodule{Foo}
Meaning:-\present\bar\
\end{module}
\begin{module}{Importtest2}
\importmodule{Importtest}
Meaning:-\present\bar\
\end{module}
```

Module 6.1.1[Foo]

Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

Meaning: >macro:->\protect \bar <

Module 6.1.2[Importtest]

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

Module 6.1.3[Importtest2]

modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Importtest Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo}<

`\usemodule` `\importmodule[⟨archive-ID⟩]{⟨module-path⟩}`

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

Test 9

```
\begin{module}{UseTest1}
\symdecl{foo}
\end{module}
\begin{module}{UseTest2}
\usemodule{UseTest1}
\symdecl{bar}
Meaning:~\present\foo\\
\end{module}
\begin{module}{UseTest3}
\importmodule{UseTest2}
Meaning:~\present\foo\\
Meaning:~\present\bar\\

All modules: \ExplSyntaxOn
\seq_use:Nn \l_stex_all_modules_seq {,~} \\
All symbols:~
\seq_use:Nn \l_stex_all_symbols_seq {,~}
\ExplSyntaxOff
\end{module}
```

Module 6.1.4[UseTest1]

Module 6.1.5[UseTest2]
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextestUseTest1 Meaning: >undefined<

Module 6.1.6[UseTest3]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2 Meaning: >undefined<
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar}<

All modules: <http://mathhub.info/sTeX?Metatheory>, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3,
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2
All symbols: <http://mathhub.info/sTeX?Metatheory?isa>, <http://mathhub.info/sTeX?Metatheory?bind>, <http://mathhub.info/sTeX?Metatheory?collect>,
<http://mathhub.info/sTeX?Metatheory?fromto>, <http://mathhub.info/sTeX?Metatheory?apply>, <http://mathhub.info/sTeX?Metatheory?collection>,
<http://mathhub.info/sTeX?Metatheory?seqtype>, <http://mathhub.info/sTeX?Metatheory?sequence-index>, <http://mathhub.info/sTeX?Metatheory?aseqfromto>, <http://mathhub.info/sTeX?Metatheory?aseqfromtovia>, <http://mathhub.info/sTeX?Metatheory?module-type>, <http://mathhub.info/sTeX?Metatheory?mathematical-structure>,
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar

Test 10

```
Circular dependencies:
\begin{module}{CircDep1}
\importmodule[Foo/Bar]{circular1?Circular1}
\importmodule[Bar/Foo]{circular2?Circular2}
\present\fooA\\
\present\fooB\\
\end{module}
```

Circular dependencies:

Module 6.1.7[CircDep1]
>macro:->\stex_invoke_symbol:n {<http://mathhub.info/tests/Foo/Bar/circular1?Circular1?fooA>}<
>macro:->\stex_invoke_symbol:n {<http://mathhub.info/tests/Bar/Foo/circular2?Circular2?fooB>}<

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {<archive-ID>} {<module-path>}`

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

If `<archive-ID>` is empty, it is automatically set to the ID of the current archive (if one exists).

1. If `<archive-ID>` is empty:

(a) If `<path>` is empty, then `<name>` must have been declared earlier in the same file and retrievable from `\g_stex_modules_in_file_seq`, or a file with name `<name>.<lang>.tex` must exist in the same folder, containing a module `<name>`. That module should have the same namespace as the current one.

(b) If `<path>` is not empty, it must point to the relative path of the containing file as well as the namespace.

2. Otherwise:

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

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

(b) If `<path>` 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.

`\stex_import_require_module:nnnn`

`{<ns>} {<archive-ID>} {<path>} {<name>}`

Checks whether a module with URI `<ns>?<name>` already exists. If not, it looks for a plausible file that declares a module with that URI.

Finally, activates that module by executing its `content`-field.

Chapter 7

STEX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

<u><code>\symdecl</code></u>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</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. 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[args=ii]{plus}` 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[args=a]{plus}` 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[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

`\stex_symdecl_do:n`

Implements the core functionality of `\symdecl`, and is called by `\symdecl` and `\symdef`.

Ultimately stores the symbol $\langle URI \rangle$ in the property list `\g_stex_symdecl_⟨URI⟩_prop` with fields:

- `name` (string),
- `module` (string),
- `notations` (sequence of strings; initially empty),
- `local` (boolean),
- `type` (token list),
- `args` (string of `is`, `as` and `bs`),
- `arity` (integer string),
- `assocs` (integer string; number of associative arguments),

Test 11

```
\begin{module}{SymdeclTest}
\symdecl[name=foo, args=3]{bar}
\symdecl[name=foobar, args=iab]{bari}
\symdecl[def=\bar* abc]{bardef}
\ExplSyntaxOn
Meaning:~\present\bar\\
\stex_get_symbol:n { bar }
Result:~\l_stex_get_symbol_uri_str\\
Meaning:~\present\bardef\\
\ExplSyntaxOff
\end{module}
```

```
Module 7.1.1[SymdeclTest]
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo}<
Result: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?foo
Meaning: >macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?SymdeclTest?bardef}<
```

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

Computes the full URI of a symbol from a macro argument, e.g. the macro name, the macro itself, the full URI...

`\notation`

`\notation[⟨args⟩]{⟨symbol⟩}{⟨notations+⟩}`

Introduces a new notation for $\langle symbol \rangle$, see `\stex_notation_do:nn`

`\stex_notation_do:nn`

`\stex_notation_do:nn{<URI>}{<notations+>}`

Implements the core functionality of `\notation`, and is called by `\notation` and `\symdef`.

Ultimately stores the notation in the property list `\g_stex_notation_<URI>#<variant>#<lang>_prop` with fields:

- symbol (URI string),
- language (string),
- variant (string),
- opprec (integer string),
- argprecs (sequence of integer strings)

Test 12

```
\begin{module}{NotationTest}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _ {#3} \comp\rangle }
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^ {#3} \comp\rangle }{ {#1}_{\comp
```

Module 7.1.2[NotationTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo

`\symdef`

`\symdef[<args>]{<symbol>}{<notations+>}`

Combines `\symdecl` and `\notation` by introducing a new symbol and assigning a new notation for it.

Test 13

```
\begin{module}{SymdefTest}
\symdef[ args=a, prec=50]{plus}{ #1 }{#1 \comp+ #2}
$\plus{a,b,c}$
\end{module}
```

Module 7.1.3[SymdefTest]
 $a+b+c$

Chapter 8

STEX-Terms

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

8.1 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/>	$\backslash\infprec$ \backslashneginfprec	Maximal and minimal notation precedences.
<hr/> <hr/>	\backslashdobrackets	$\backslashdobrackets \{ \langle body \rangle \}$ Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S I E X brackets (by default (and)), which can be changed temporarily using $\backslash\withbrackets$.
<hr/> <hr/>	$\backslash\withbrackets$	$\backslash\withbrackets \langle left \rangle \langle right \rangle \{ \langle body \rangle \}$ Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \S I E X 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 $\backslash\left$ and $\backslash\right$ in display-mode.

Test 14

```

\begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _{#3} \comp\rangle }
$\bar{abc}$ and $\bar{foo}[abc]$.
\end{module}

```

Module 8.1.1[MathTest1]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle a^b_c \rangle$
and $\langle a^b_c \rangle$.

Test 15

```

\begin{module}{MathTest2}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{foobar}{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle }{ {#1}_{\comp\langle #1 \comp\mid [ #2 ] ^{#3} \comp\rangle } }
$\foobar{a\{b,c,d,e,f\}g}$ and $\foobar[foo]{a\{b,c\}g}$ and $\foobar{abc}$

\symdecl[ args=a]{ plus }
\symdecl[ args=a]{ mult }
\notation[prec=50]{ plus }{#1}{#1 \comp+ #2}
\notation[prec=100]{ mult }{#1}{#1 \comp\cdot #2}
$\plus{a,\mult{b,c}}$ and $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$
$[\plus{a,\mult{b,c}}]\text{ and }[\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}]$
$\displaystyle \plus{a,\mult{b,c}}$ and
\withbrackets[{$\displaystyle
\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$}
\end{module}

```

Module 8.1.2[MathTest2]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo $\langle a|[b:c,d:e,f]^9 \rangle$
and $\langle a|[b:c]^9 \rangle$ and $\langle a|[b]^c \rangle$
 $a+(b \cdot c)$ and $a \cdot \frac{a}{b} + \frac{a}{c}$
 $a+(b \cdot c)$ and $a \cdot \frac{a}{b} + \frac{a}{c}$

`\stex_term_custom:nn`

`\stex_term_custom:nn{<URI>}{<args>}`

Implements custom one-time notation. Invoked by `\stex_invoke_symbol:n` in text mode, or if followed by `*` in math mode, or whenever followed by `!`.

Test 16

```
\begin{module}{TextTest}
\importmodule{Foo}

\bar[some ]a[ and some ]b[ and also some ]c[ here].

$\bar*[\text{some }]a[\text{ and some }]b[\text{ and also some }]c[\text{ here}]\$.

$\bar!![\mathtt{bar}]\$

\bar*{a}*{b}[or just some ]c

\bar![bar]

\bar[or first ]*[2]{b}[ , then ]*[3]{c}[ , and finally ]a

\end{module}
```

```
Module 8.1.3[TextTest]
modulesImporting module: file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo
some aand some band also some here.
some a and some b and also some c here.
bar
or just some c
bar
or first b, then c, and finally a
```

`\stex_highlight_term:nn`

`\stex_highlight_term:nn{<URI>}{<args>}`

Establishes a context for `\comp`. Stores the URI in a variable so that `\comp` knows which symbol governs the current notation.

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

Marks `<args>` as a notation component of the current symbol for highlighting, linking, etc.

The precise behavior is governed by `\@comp`, which takes as additional argument the URI of the current symbol. By default, `\@comp` adds the URI as a PDF tooltip and colors the highlighted part in blue.

`\@defemph` behaves like `\@comp`, and can be similarly redefined, but marks an expression as *definiendum* (used by `\definiendum`)

`\STEXinvisible`

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.

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

mathstructure TODO

Test 17

```

\begin{module}{StructureTest1}
\begin{mathstructure}[name=Magma]{magma}
\symdef{universe}{\comp M}
\symdef[ args=2]{op}{#1 \comp\circ #2}
$isa{\op ab}{universe$
\end{mathstructure}

\ExplSyntaxOn
\prop_get:NnN \g_stex_last_feature__prop {fields} \l_tmpa_seq
\seq_use:Nn \l_tmpa_seq {,}
\ExplSyntaxOff

\present\magma

\instantiate{magma}[
universe ! {{\comp U}},
op ! {{#1 \comp+ #2 }}
]{mM}
\notation[op = U]{mM/universe}{\comp U}
\notation[op = +]{mM/op}{#1 \comp+ #2}

Test: $\mM{op}ab$

Test2: $\mM{}$
\end{module}

```

```

Module 9.1.1[StructureTest1]
aob:M
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}
feature?op
>\macro->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}
Test: a+b
Test2: (U,+)

```

Chapter 10

TeX-Statements

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

10.1 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 11

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

11.1 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 it's 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,for=sum-over-odds]
  {We prove that  $\sum_{i=1}^n (2i-1) = n^2$  by induction over  $n$ }
  \begin{spfcase}{For the induction we have to consider the following cases:}
    \begin{spfcase}{ $n=1$ }
      \begin{spfstep}[display=flow] then we compute  $1=1^2$ \end{spfstep}
    \end{spfcase}
    \begin{spfcase}{ $n=2$ }
      \begin{sproofcomment}[display=flow]
        This case is not really necessary, but we do it for the
        fun of it (and to get more intuition).
      \end{sproofcomment}
      \begin{spfstep}[display=flow] 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}
    \begin{spfstep}[type=conclusion]
      We have considered all the cases, so we have proven the assertion.
    \end{spfstep}
  \end{spfcase}
\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).⁴

⁴EdNOTE: talk a bit more about proofs and their structure,... maybe copy from OMDoc spec.

11.2 The User Interface

11.2.1 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).

11.2.2 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.

`spfidea`

`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.

11.2.3 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	
P.1	For the induction we have to consider the following cases:	
P.1.1	$n = 1$: then we compute $1 = 1^2$	□
P.1.1	$n = 2$: This case is not really necessary, but we do it for the fun of it (and to get more intuition). We compute $1 + 3 = 2^2 = 4$	□
P.1.1	$n > 1$:	
P.1.1.1	Now, we assume that the assertion is true for a certain $k \geq 1$, i.e. $\sum_{i=1}^k (2i - 1) = k^2$.	
P.1.1.1	We have to show that we can derive the assertion for $n = k + 1$ from this assumption, i.e. $\sum_{i=1}^{k+1} (2i - 1) = (k + 1)^2$.	
P.1.1.1	We obtain $\sum_{i=1}^{k+1} (2i - 1) = \sum_{i=1}^k (2i - 1) + 2(k + 1) - 1$ by splitting the sum	
P.1.1.1	Thus we have $\sum_{i=1}^{k+1} (2i - 1) = k^2 + 2k + 1$ by inductive hypothesis.	
P.1.1.1	We can simplify the right-hand side to $(k + 1)^2$, which proves the assertion.	□
P.1.1	We have considered all the cases, so we have proven the assertion.	□

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

11.2.4 Proof Structure

subproof	The pfcases environment is used to mark up a subproof. This environment takes an optional KeyVal argument for semantic annotations and a second argument that allows to specify an introductory comment (just like in the proof environment). The method key can be used to give the name of the proof method executed to make this subproof.
spfcases	The pfcases environment is used to mark up a proof by cases. Technically it is a variant of the subproof where the method is by-cases . Its contents are spfcases environments that mark up the cases one by one.
spfcases	The content of a pfcases environment are a sequence of case proofs marked up in the pfcases environment, which takes an optional KeyVal argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a pfcases environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcases environment that takes the same arguments, but instead of the spfsteps in the body uses a third argument for a proof sketch.
\spfcasesketch	
sproofcomment	The proofcomment environment is much like a step , 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 \premise .

11.2.5 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 `sproof` package provides the `\sproofend` macro for this. If a different symbol for the proof end is to be used (e.g. *q.e.d*), then this can be obtained by specifying it using the `\sProofEndSymbol` configuration macro (e.g. by specifying `\sProofEndSymbol{q.e.d}`).

`\sProofEndSymbol`

Some of the proof structuring macros above will insert proof end symbols for sub-proofs, 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 `proofend={}` in them or use `\sProofEndSymbol{}`.

11.2.6 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.⁵ The proof step labels can be customized via the `\pstlabelstyle` macro:

EdN:5

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{<style>}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@<style>` that takes 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 L^AT_EX `\@for...:=... \do{...}` macro; see Figure 2 for examples.

style	example	configuration macro
<code>long</code>	<code>0.8.1.5</code>	<code>\def\pst@make@label@long#1#2{\@for\@I:=#1\do{\@I.}#2}</code>
<code>angles</code>	<code>>>>5</code>	<code>\def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}</code>
<code>short</code>	<code>5</code>	<code>\def\pst@make@label@short#1#2{#2}</code>
<code>empty</code>		<code>\def\pst@make@label@empty#1#2{}</code>

Figure 2: Configuration Proof Step Label Styles

11.3 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 S_TE_X issue tracker at [\[sTeX\]](#).

⁵EdNOTE: we might want to develop an extension `sproof-babel` in the future.

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 12

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.

12.1 Symbols

Part III
Extensions

Chapter 13

Tikzinput

13.1 Macros and Environments

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

Chapter 14

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

The `omdoc` 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.

14.1 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 `omdoc` 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

14.2 The User Interface

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

14.2.1 Package and Class Options

The `omdoc` 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 `omdoc` package accepts the same except the first two.

14.2.2 Document Structure

document

\documentkeys

id

The top-level `document` environment can be given key/value information by the `\documentkeys` macro in the preamble². This can be used to give metadata about the document. For the moment only the `id` key is used to give an identifier to the `omdoc` element resulting from the \LaTeX ML transformation.

omgroup

id

creators

contributors

short

loadmodules

The structure of the document is given by the `omgroup` environment just like in `OMDOC`. In the \LaTeX route, the `omgroup` environment is flexibly mapped to sectioning commands, inducing the proper sectioning level from the nesting of `omgroup` environments. Correspondingly, the `omgroup` environment takes an optional key/value argument for metadata followed by a regular argument for the (section) title of the `omgroup`. The optional metadata argument has the keys `id` for an identifier, `creators` and `contributors` for the Dublin Core metadata [DCM03]; see [Koh20a] for details of the format. The `short` allows to give a short title for the generated section. If the title contains semantic macros, they need to be protected by `\protect`, and we need to give the `loadmodules` key it needs no value. For instance we would have

```

\begin{module}{foo}
\symdef{bar}{B^a_r}
...
\begin{omgroup}[id=sec.barderiv,loadmodules]{Introducing $\protect\bar$ Derivations}

```

blindomgroup

\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 `omdoc` package provides a variant `blindomgroup`

⁶EDNOTE: integrate with latexml's XMRef in the Math mode.

²We cannot patch the document environment to accept an optional argument, since other packages we load already do; pity.

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 frontmatter³ 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{blindomgroup}
\begin{blindomgroup}
\begin{frontmatter}
\maketitle\newpage
\begin{omgroup}[display=flow]{Preface}
... <<preface>> ...
\end{omgroup}
\clearpage\setcounter{tocdepth}{4}\tableofcontents\clearpage
\end{frontmatter}
\end{blindomgroup}
... <<introductory remarks>> ...
\end{blindomgroup}
\begin{omgroup}{Introduction}
... <<intro>> ...
\end{omgroup}
... <<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.

14.2.3 Ignoring Inputs

`ignore`
`showignores`

The `ignore` environment can be used for hiding text parts from the document structure. The body of the environment is not PDF or DVI output unless the `showignores` option

³We shied away from redefining the `frontmatter` to induce a `blindomgroup`, but this may be the “right” way to go in the future.

is given to the `omdoc` 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`.

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.

`\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].

14.2.4 Structure Sharing

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.

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.⁷

14.2.5 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.

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

⁷EdNOTE: document LMID und LMXRef here if we decide to keep them.

14.2.6 Colors

For convenience, the `omdoc` 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.

14.3 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 15

Slides and Course Notes

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

15.1 Introduction

The `mikoslides` 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`.

15.2 The User Interface

The `mikoslides` class takes the notion of a slide frame from Till Tantau’s excellent `beamer` class and adapts its notion of frames for use in the \TeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `mikoslides` 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 `mikoslides` class has two modes: *slides mode* and *notes mode* which are determined by the package option.

15.2.1 Package Options

The `mikoslides` class takes a variety of class options:⁸

- | | |
|---|--|
| <code>slides</code>
<code>notes</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 15.2.2). |
| <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 15.2.4). 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>.

15.2.2 Notes and Slides

`frame` Slides are represented with the `frame` just like in the `beamer` class, see [Tanb] for details.
`note` The `mikoslides` class adds the `note` environment for encapsulating the course note fragments.⁴

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

⁸EDNOTE: leaving out `noproblems` for the moment until we decide what to do with it.

⁴MK: 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.

⚠: 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 `mikoslides` 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 `nomtext` environment is just an `omtext` 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.

15.2.3 Header and Footer Lines of the Slides

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

The default footer line of the `mikoslides` 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 `mikoslides` 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.

15.2.4 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 \LaTeX 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.⁹

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}
```


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

⁹EdNOTE: MK: the `hyperref` link does not seem to work yet. I wonder why but do not have the time to fix it.

`\mhframeimage{baz/foobar}`

15.2.5 Colors and Highlighting

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

15.2.6 Front Matter, Titles, etc.

15.2.7 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{nomtext}[title=Excursion]
                  \activateexcursion{founif}{../ex/founif}
                  We will cover first-order unification in \sref{founif}.
                  \end{nomtext}
```

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

15.2.8 Miscellaneous

15.3 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. 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 16

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.

16.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. 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.

16.2 The User Interface

16.2.1 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).

⁵for the moment multiple choice problems are not supported, but may well be in a future version

16.2.2 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{problem}[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{problem}
\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).

```
Problem0.0 ()
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.

16.2.3 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

16.2.4 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.

16.2.5 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.

16.3 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{problem}[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{problem}

```

Problem0.0 ()

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

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

Problem0.0 ()

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 17

`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

17.1 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.

17.2 The User Interface

17.2.1 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].

17.2.2 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).

17.2.3 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`

17.2.4 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.

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 `STEX`GitHub repository [\[sTeX\]](#).

1. none reported yet.

Part IV
Implementation

Chapter 18

STEX -Basics Implementation

18.1 The STEXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{2021/08/01}{1.9}{bla}
7 \LoadClass[border=1px,varwidth]{standalone}
8 \setlength\textwidth{15cm}
9
10 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{stex}}
11 \ProcessOptions
12
13 \RequirePackage{stex}
14 </cls>
```

18.2 Preliminaries

```
15 <*package>
16
17 %%%%%%%%% basics.dtx %%%%%%%%%
18
19 \RequirePackage{expl3,l3keys2e,ltxcmds}
20 \ProvidesExplPackage{stex}{2021/08/01}{1.9}{bla}
21 \RequirePackage{expl-keystr-compatible}
22 \RequirePackage{morewrites}
23
24 Package options:
25 \keys_define:nn { stex } {
26   debug      .clist_set:N = \c_stex_debug_clist ,
27   showmods   .bool_set:N  = \c_stex_showmods_bool ,
```

```

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

```

\stex The \TeX logo:

\sTeX

```

33 \protected\def\stex{%
34   \@ifundefined{texorpdfstring}%
35   {\let\texorpdfstring\@firstoftwo}%
36   }%
37   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
38 }
39 \def\sTeX{\stex}

```

(End definition for `\stex` and `\sTeX`. These functions are documented on page 9.)

18.3 Messages and logging

```

40 <@@=stex_log>

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

```

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

```

51 \cs_new_protected:Nn \stex_debug:nn {
52   \clist_if_in:NnTF \c_stex_debug_clist { all } {
53     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
54       \Debug~#1:~#2\\
55     }
56     \msg_none:nn{stex}{debug / #1}
57   }{
58     \clist_if_in:NnTF \c_stex_debug_clist { #1 } {
59       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
60         \Debug~#1:~#2\\
61       }
62       \msg_none:nn{stex}{debug / #1}
63     }
64   }
65 }

```

(End definition for `\stex_debug:nn`. This function is documented on page 9.)

Redirecting messages:

```

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

```

18.4 Persistence

75 $\langle @@=stex_persist \rangle$

$\backslash c_stex_persist_sms_iow$ File variable used for the sms-File

```

76 \iow_new:N \c__stex_persist_sms_iow
77 \AddToHook{begindocument}{
78   \bool_if:NTF \c_stex_persist_mode_bool {
79     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
80   } {
81     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
82   }
83 }
84 \AddToHook{enddocument}{
85   \bool_if:NF \c_stex_persist_mode_bool {
86     \iow_close:N \c__stex_persist_sms_iow
87   }
88 }

```

(End definition for $\backslash c_stex_persist_sms_iow$.)

$\backslash stex_add_to_sms:n$ Adds the provided code to the .sms-file of the document.

```

89 \cs_new_protected:Nn \stex_add_to_sms:n {
90   \bool_if:NF \c_stex_persist_mode_bool {
91     \iow_now:Nn \c__stex_persist_sms_iow { #1 }
92   }
93 }

```

(End definition for $\backslash stex_add_to_sms:n$. This function is documented on page 9.)

18.5 HTML Annotations

94 $\langle @@=stex_annotate \rangle$
95 $\backslash RequirePackage\{rustex\}$

We add the namespace abbreviation $ns:stex="http://kwarc.info/ns/sTeX"$ to $RusT_{E}X$:

```

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

```

$\backslash if@latexml$ Conditionals for L^AT_EX_ML:

```

\latexml_if_p:
\latexml_if:TF
97 \ifcsname if@latexml\endcsname\else
98   \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
99 \fi

```

```

100
101 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
102   \if@latexml
103     \prg_return_true:
104   \else:
105     \prg_return_false:
106   \fi:
107 }

```

(End definition for \if@latexml and \latexml_if:TF. These functions are documented on page 9.)

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

```

\c__stex_annotate_emptyarg_tl
108 \tl_new:N \l__stex_annotate_arg_tl
109 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
110   \rustex_if:TF {
111     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
112   }{-}
113 }

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

```

\__stex_annotate_checkempty:n
114 \cs_new_protected:Nn \__stex_annotate_checkempty:n {
115   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
116   \tl_if_empty:NT \l__stex_annotate_arg_tl {
117     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
118   }
119 }

```

(End definition for __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool Whether to (locally) produce HTML output

```

\stex_if_do_html:
120 \bool_new:N \l_stex_html_do_output_bool
121 \bool_set_true:N \l_stex_html_do_output_bool
122 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
123   \bool_if:nTF \l_stex_html_do_output_bool
124     \prg_return_true: \prg_return_false:
125 }

```

(End definition for \l_stex_html_do_output_bool and \stex_if_do_html:. These functions are documented on page ??.)

\stex_suppress_html:n Whether to (locally) produce HTML output

```

126 \cs_new_protected:Nn \stex_suppress_html:n {
127   \exp_args:Nne \use:nn {
128     \bool_set_false:N \l_stex_html_do_output_bool
129     #1
130   }{
131     \stex_if_do_html:T {
132       \bool_set_true:N \l_stex_html_do_output_bool
133     }
134   }
135 }

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

185 \par\rustex_annotate_HTML_end:
186 }
187 }{
188 \latexml_if:TF {
189 \cs_new_protected:Nn \stex_annotate:nnn {
190 \__stex_annotate_checkempty:n { #3 }
191 \mode_if_math:TF {
192 \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
193 \tl_use:N \l__stex_annotate_arg_tl
194 }
195 }{
196 \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
197 \tl_use:N \l__stex_annotate_arg_tl
198 }
199 }
200 }
201 \cs_new_protected:Nn \stex_annotate_invisible:n {
202 \__stex_annotate_checkempty:n { #1 }
203 \mode_if_math:TF {
204 \cs:w latexml@invisible@math\cs_end:{
205 \tl_use:N \l__stex_annotate_arg_tl
206 }
207 } {
208 \cs:w latexml@invisible@text\cs_end:{
209 \tl_use:N \l__stex_annotate_arg_tl
210 }
211 }
212 }
213 \cs_new_protected:Nn \stex_annotate_invisible:nnn {
214 \__stex_annotate_checkempty:n { #3 }
215 \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
216 \tl_use:N \l__stex_annotate_arg_tl
217 }
218 }
219 \NewDocumentEnvironment{stex_annotate_env} { m m } {
220 \par\begin{latexml@annotateenv}{#1}{#2}
221 }{
222 \par\end{latexml@annotateenv}
223 }
224 }{
225 \cs_new_protected:Nn \stex_annotate:nnn {#3}
226 \cs_new_protected:Nn \stex_annotate_invisible:n {}
227 \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
228 \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
229 }
230 }

```

(End definition for `\stex_annotate:nnn`, `\stex_annotate_invisible:n`, and `\stex_annotate_invisible:nnn`.
These functions are documented on page [10](#).)

18.6 Languages

```

231 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

232 \prop_const_from_keyval:Nn \c_stex_languages_prop {
233   en = english ,
234   de = ngerman ,
235   ar = arabic ,
236   bg = bulgarian ,
237   ru = russian ,
238   fi = finnish ,
239   ro = romanian ,
240   tr = turkish ,
241   fr = french
242 }
243
244 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
245   english   = en ,
246   ngerman   = de ,
247   arabic    = ar ,
248   bulgarian = bg ,
249   russian   = ru ,
250   finnish   = fi ,
251   romanian  = ro ,
252   turkish   = tr ,
253   french    = fr
254 }
255 % todo: chinese simplified (zhs)
256 %       chinese traditional (zht)

```

(End definition for `\c_stex_languages_prop` and `\c_stex_language_abbrevs_prop`. These variables are documented on page 10.)

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

```

257 \clist_if_empty:NF \c_stex_languages_clist {
258   \clist_clear:N \l_tmpa_clist
259   \clist_map_inline:Nn \c_stex_languages_clist {
260     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
261       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
262     } {
263       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
264     }
265   }
266   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
267   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
268 }

```

18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```


(End definition for \stex_deactivate_macro:Nn. This function is documented on page 10.)

\stex_reactivate_macro:N

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

(End definition for \stex_reactivate_macro:N. This function is documented on page 10.)

```
278 \</package>
```

Chapter 19

STEX -MathHub Implementation

```
279 <*package>
280
281 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
282
283 <@@=stex_path>
284
285 Warnings and error messages
286 \msg_new:nnn{stex}{error/norepository}{
287   No~archive~#1~found~in~#2
288 }
289 \msg_new:nnn{stex}{error/notinarchive}{
290   Not~currently~in~an~archive,~but~\detokenize{#1}~
291   needs~one!
292 }
293 \msg_new:nnn{stex}{error/nofile}{
294   \detokenize{#1}~could~not~find~file~#2
295 }
```

19.1 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
\stex_path_from_string:NV
\stex_path_from_string:cn
\stex_path_from_string:cV
294 \cs_new_protected:Nn \stex_path_from_string:Nn {
295   \str_set:Nx \l_tmpa_str { #2 }
296   \str_if_empty:NTF \l_tmpa_str {
297     \seq_clear:N #1
298   }{
299     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
300     \sys_if_platform_windows:T{
301       \seq_clear:N \l_tmpa_tl
302       \seq_map_inline:Nn #1 {
303         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
304         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

305     }
306     \seq_set_eq:NN #1 \l_tmpa_tl
307   }
308   \stex_path_canonicalize:N #1
309 }
310 }
311 \cs_generate_variant:Nn \stex_path_from_string:Nn
312 { NV, cn, cV }

```

(End definition for `\stex_path_from_string:Nn`. This function is documented on page 11.)

```

\stex_path_to_string:NN
\stex_path_to_string:N
313 \cs_new_protected:Nn \stex_path_to_string:NN {
314   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
315 }
316
317 \cs_new:Nn \stex_path_to_string:N {
318   \seq_use:Nn #1 /
319 }

```

(End definition for `\stex_path_to_string:NN` and `\stex_path_to_string:N`. These functions are documented on page 11.)

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
320 \str_const:Nn \c__stex_path_dot_str {.}
321 \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.

```

322 \cs_new_protected:Nn \stex_path_canonicalize:N {
323   \seq_if_empty:NF #1 {
324     \seq_clear:N \l_tmpa_seq
325     \seq_get_left:NN #1 \l_tmpa_tl
326     \str_if_empty:NT \l_tmpa_tl {
327       \seq_put_right:Nn \l_tmpa_seq {}
328     }
329     \seq_map_inline:Nn #1 {
330       \str_set:Nn \l_tmpa_tl { ##1 }
331       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
332         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
333           \seq_if_empty:NTF \l_tmpa_seq {
334             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
335               \c__stex_path_up_str
336             }
337           }{
338             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
339             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
340               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
341                 \c__stex_path_up_str
342               }
343             }{
344               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
345             }

```

```

346     }
347   }{
348     \str_if_empty:NF \l_tmpa_tl {
349       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
350     }
351   }
352 }
353 }
354 \seq_gset_eq:NN #1 \l_tmpa_seq
355 }
356 }

```

(End definition for `\stex_path_canonicalize:N`. This function is documented on page 11.)

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

```

357 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
358   \seq_if_empty:NTF #1 {
359     \prg_return_false:
360   }{
361     \seq_get_left:NN #1 \l_tmpa_tl
362     \str_if_empty:NTF \l_tmpa_tl {
363       \prg_return_true:
364     }{
365       \prg_return_false:
366     }
367   }
368 }

```

(End definition for `\stex_path_if_absolute:NTF`. This function is documented on page 11.)

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

369 \str_new:N\l_stex_kpsewhich_return_str
370 \cs_new_protected:Nn \stex_kpsewhich:n {
371   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
372   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
373   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
374 }

```

(End definition for `\stex_kpsewhich:n`. This function is documented on page 11.)

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

375 \sys_if_platform_windows:TF{
376   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
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 11.)

19.3 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 11.)

`\g_stex_currentfile_seq` Hooks for file inputs that push/pop `\g__stex_files_stack` to update `\c_stex_mainfile_seq`.

```

389 \seq_gclear_new:N\g_stex_currentfile_seq
390 \AddToHook{file/before}{
391   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
392   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
393     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
394   }{
395     \stex_path_from_string:Nn\g_stex_currentfile_seq{
396       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
397     }
398   }
399   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
400   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
401 }
402 \AddToHook{file/after}{
403   \seq_if_empty:NF\g__stex_files_stack{
404     \seq_gpop:Nn\g__stex_files_stack\l_tmpa_seq
405   }
406   \seq_if_empty:NTF\g__stex_files_stack{
407     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
408   }{
409     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
410     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
411   }
412 }

```

(End definition for `\g_stex_currentfile_seq`. This variable is documented on page 12.)

19.4 MathHub Repositories

```

413 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
414 \str_if_empty:NTF\mathhub{
415   \stex_kpsewhich:n{-var-value~MATHHUB}
416   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
417
418   \str_if_empty:NTF\c_stex_mathhub_str{
419     \msg_warning:nn{stex}{warning/nomathhub}
420   }{
421     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
422     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
423   }
424 }{
425   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
426   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
427     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
428       \c_stex_pwd_str/\mathhub
429     }
430   }
431   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
432   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
433 }

```

(End definition for `\mathhub`, `\c_stex_mathhub_seq`, and `\c_stex_mathhub_str`. These variables are documented on page 12.)

```

\__stex_mathhub_do_manifest:n
434 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
435   \str_set:Nx \l_tmpa_str { #1 }
436   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
437     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
438     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
439     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
440     \__stex_mathhub_find_manifest:N \l_tmpa_seq
441     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
442       \msg_error:nnnn{stex}{error/norepository}{#1}{
443         \stex_path_to_string:N \c_stex_mathhub_str
444       }
445     } {
446       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
447     }
448   }
449 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
450 \str_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`:

```

451 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
452   \seq_set_eq:NN \l_tmpa_seq #1
453   \bool_set_true:N \l_tmpa_bool
454   \bool_while_do:Nn \l_tmpa_bool {
455     \seq_if_empty:NTF \l_tmpa_seq {
456       \bool_set_false:N \l_tmpa_bool
457     }{
458       \file_if_exist:nTF{
459         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
460       }{
461         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
462         \bool_set_false:N \l_tmpa_bool
463       }{
464         \file_if_exist:nTF{
465           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
466         }{
467           \seq_put_right:Nn \l_tmpa_seq{META-INF}
468           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
469           \bool_set_false:N \l_tmpa_bool
470         }{
471           \file_if_exist:nTF{
472             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
473           }{
474             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
475             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
476             \bool_set_false:N \l_tmpa_bool
477           }{
478             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
479           }
480         }
481       }
482     }
483   }
484   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
485 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

486 \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:

```

487 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
488   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
489   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
490   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
491     \str_set:Nn \l_tmpa_str {##1}
492     \exp_args:NNoo \seq_set_split:Nnn
493       \l_tmpb_seq \c_colon_str \l_tmpa_str
494     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

495 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
496 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
497 }
498 \exp_args:No \str_case:nnTF \l_tmpa_tl {
499 {id} {
500 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
501 { id } \l_tmpb_tl
502 }
503 {narration-base} {
504 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
505 { narr } \l_tmpb_tl
506 }
507 {url-base} {
508 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
509 { docurl } \l_tmpb_tl
510 }
511 {source-base} {
512 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
513 { ns } \l_tmpb_tl
514 }
515 {ns} {
516 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
517 { ns } \l_tmpb_tl
518 }
519 {dependencies} {
520 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
521 { deps } \l_tmpb_tl
522 }
523 }{}{}
524 }{}
525 }
526 \ior_close:N \c__stex_mathhub_manifest_ior
527 }

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

528 \cs_new_protected:Nn \stex_set_current_repository:n {
529 \stex_require_repository:n { #1 }
530 \prop_set_eq:Nc \l_stex_current_repository_prop {
531 c_stex_mathhub_#1_manifest_prop
532 }
533 }

```

(End definition for `\stex_set_current_repository:n`. This function is documented on page 13.)

`\stex_require_repository:n`

```

534 \cs_new_protected:Nn \stex_require_repository:n {
535 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
536 \stex_debug:nn{mathhub}{Opening~archive:~#1}
537 \__stex_mathhub_do_manifest:n { #1 }
538 \exp_args:Nx \stex_add_to_sms:n {
539 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
540 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
541 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```



```

542     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
543     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
544   }
545 }
546 }
547 }

```

(End definition for `\stex_require_repository:n`. This function is documented on page 13.)

`\l_stex_current_repository_prop` Current MathHub repository

```

548 \prop_new:N \l_stex_current_repository_prop
549
550 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
551 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
552   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
553 } {
554   \__stex_mathhub_parse_manifest:n { main }
555   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
556   \l_tmpa_str
557   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
558   \c_stex_mathhub_main_manifest_prop
559   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
560   \stex_debug:nn{mathhub}{Current~repository:~
561     \prop_item:Nn \l_stex_current_repository_prop {id}
562   }
563 }

```

(End definition for `\l_stex_current_repository_prop`. This variable is documented on page 12.)

`\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.

```

564 \cs_new_protected:Nn \stex_in_repository:nn {
565   \str_set:Nx \l_tmpa_str { #1 }
566   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
567   \str_if_empty:NTF \l_tmpa_str {
568     \exp_args:Ne \l_tmpa_cs{
569       \prop_item:Nn \l_stex_current_repository_prop { id }
570     }
571   }{
572     \stex_require_repository:n \l_tmpa_str
573     \str_set:Nx \l_tmpa_str { #1 }
574     \exp_args:Nne \use:nn {
575       \stex_set_current_repository:n \l_tmpa_str
576       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
577     }{
578       \stex_set_current_repository:n {
579         \prop_item:Nn \l_stex_current_repository_prop { id }
580       }
581     }
582   }
583 }

```

(End definition for `\stex_in_repository:nn`. This function is documented on page 13.)

```

\inputref
\stex_inputref:nn
584 \newif \ifinputref \inputreffalse
585
586 \cs_new_protected:Nn \stex_inputref:nn {
587   \stex_in_repository:nn {#1} {
588     \ifinputref
589       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
590     \else
591       \inputreftrue
592       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
593     \inputreffalse
594   \fi
595 }
596 }
597 \NewDocumentCommand \inputref { 0{} m}{
598   \stex_inputref:nn{ #1 }{ #2 }
599 }
600
601 \cs_new_protected:Nn \stex_mhbibresource:nn {
602   \stex_in_repository:nn {#1} {
603     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
604   }
605 }
606 \newcommand\addmhbibresource[2][]{
607   \stex_mhbibresource:nn{ #1 }{ #2 }
608 }

```

(End definition for `\inputref` and `\stex_inputref:nn`. These functions are documented on page 13.)

`\mhpath`

```

609 \def \mhpath #1 #2 {
610   \exp_args:Ne \str_if_eq:nnTF{#1}{#{
611     \c_stex_mathhub_str /
612     \prop_item:Nn \l_stex_current_repository_prop { id }
613     / source / #2
614   }{
615     \c_stex_mathhub_str / #1 / source / #2
616   }
617 }

```

(End definition for `\mhpath`. This function is documented on page 13.)

`\libinput`

```

618 \cs_new_protected:Npn \libinput #1 {
619   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
620     \msg_error:nnn{stex}{error/notinarchive}\libinput
621   }
622   \bool_set_false:N \l_tmpa_bool
623   \tl_clear:N \l_tmpa_tl
624   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
625   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
626   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
627   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
628     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

629 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
630 / meta-inf / lib / #1.tex}{
631 \bool_set_true:N \l_tmpa_bool
632 \tl_put_right:Nx \l_tmpa_tl {
633 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
634 / meta-inf / lib / #1.tex}
635 }
636 }{}
637 }
638 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
639 / \l_tmpa_str / lib / #1.tex
640 }{
641 \bool_set_true:N \l_tmpa_bool
642 \tl_put_right:Nx \l_tmpa_tl {
643 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
644 / \l_tmpa_str / lib / #1.tex}
645 }
646 }{}
647 \bool_if:NF \l_tmpa_bool {
648 \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
649 }
650 \l_tmpa_tl
651 }

```

(End definition for \libinput. This function is documented on page 13.)

```

652 </package>

```

Chapter 20

STEX -References Implementation

```
653 <*package>
654
655 %%%%%%%%%% references.dtx %%%%%%%%%%
656
657 %\RequirePackage{hyperref}
658 %\RequirePackage{cleveref}
659 <@@=stex_refs>
660
661 Warnings and error messages
662
663 \iow_new:N \c__stex_refs_refs_iow
664 \AddToHook{begindocument}{
665   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
666 }
667 \AddToHook{enddocument}{
668   \iow_close:N \c__stex_refs_refs_iow
669 }
670
671 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
672
673 \NewDocumentCommand \STEXreftitle { m } {
674   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
675 }
```

20.1 Document URIs and URLs

```
674 \seq_new:N \g__stex_refs_all_refs_seq
675
676 \str_new:N \l_stex_current_docns_str
677
678 \cs_new_protected:Nn \stex_get_document_uri: {
679   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
680   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
681   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
682   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

683 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
684
685 \str_clear:N \l_tmpa_str
686 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
687   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
688 }
689
690 \str_if_empty:NTF \l_tmpa_str {
691   \str_set:Nx \l_stex_current_docns_str {
692     file:/\stex_path_to_string:N \l_tmpa_seq
693   }
694 }{
695   \bool_set_true:N \l_tmpa_bool
696   \bool_while_do:Nn \l_tmpa_bool {
697     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
698     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
699       {source} { \bool_set_false:N \l_tmpa_bool }
700     }{}{
701       \seq_if_empty:NT \l_tmpa_seq {
702         \bool_set_false:N \l_tmpa_bool
703       }
704     }
705   }
706
707   \seq_if_empty:NTF \l_tmpa_seq {
708     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
709   }{
710     \str_set:Nx \l_stex_current_docns_str {
711       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
712     }
713   }
714 }
715 }
716
717 \str_new:N \l_stex_current_docurl_str
718 \cs_new_protected:Nn \stex_get_document_url: {
719   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
720   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
721   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
722   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
723   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
724
725   \str_clear:N \l_tmpa_str
726   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
727     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
728       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
729     }
730   }
731
732   \str_if_empty:NTF \l_tmpa_str {
733     \str_set:Nx \l_stex_current_docurl_str {
734       file:/\stex_path_to_string:N \l_tmpa_seq
735     }
736   }{
737     \bool_set_true:N \l_tmpa_bool

```

```

737 \bool_while_do:Nn \l_tmpa_bool {
738   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
739   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
740     {source} { \bool_set_false:N \l_tmpa_bool }
741   }{}{
742     \seq_if_empty:NT \l_tmpa_seq {
743       \bool_set_false:N \l_tmpa_bool
744     }
745   }
746 }
747
748 \seq_if_empty:NTF \l_tmpa_seq {
749   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
750 }{
751   \str_set:Nx \l_stex_current_docurl_str {
752     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
753   }
754 }
755 }
756 }

```

20.2 Setting Reference Targets

```

757 \str_const:Nn \c__stex_refs_url_str{URL}
758 \str_const:Nn \c__stex_refs_ref_str{REF}
759 % @currentlabel -> number
760 % @currentlabelname -> title
761 % @currentHref -> name.number <- id of some kind
762 % \theH# -> \arabic{section}
763 % \the# -> number
764 % \hyper@makecurrent{#}
765 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
766   \stex_get_document_uri:
767   \str_set:Nx \l_tmpa_str { #1 }
768   \str_if_empty:NT \l_tmpa_str {
769     \int_zero:N \l_tmpa_int
770     \bool_set_true:N \l_tmpa_bool
771     \bool_while_do:Nn \l_tmpa_bool {
772       \cs_if_exist:cTF {
773         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
774       }{
775         \int_incr:N \l_tmpa_int
776       }{
777         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
778         \bool_set_false:N \l_tmpa_bool
779       }
780     }
781   }
782   \str_set:Nx \l_tmpa_str {
783     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
784   }
785   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
786   \stex_if_smsmode:TF {
787     \stex_get_document_url:

```

```

788 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
789 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
790 }{
791 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}{
792 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
793 \str_gset:cn {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
794 }
795 }

796 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
797 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
798 }

```

20.3 Using References

```

799 \str_new:N \l__stex_refs_indocument_str
800 \keys_define:nn { stex / sref } {
801 linktext .tl_set:N = \l__stex_refs_linktext_tl ,
802 fallback .tl_set:N = \l__stex_refs_fallback_tl ,
803 pre .tl_set:N = \l__stex_refs_pre_tl ,
804 post .tl_set:N = \l__stex_refs_post_tl ,
805 %indoc .str_set_x:N = \l__stex_refs_repo_str ,
806 }
807
808 \bool_new:N \c__stex_refs_hyperref_bool
809 \bool_set_false:N \c__stex_refs_hyperref_bool
810 \AddToHook{begindocument}{
811 \ifpackageloaded{hyperref}{
812 \bool_set_true:N \c__stex_refs_hyperref_bool
813 }{}
814 }
815
816
817 \cs_new_protected:Nn \__stex_refs_args:n {
818 \tl_clear:N \l__stex_refs_linktext_tl
819 \tl_clear:N \l__stex_refs_fallback_tl
820 \tl_clear:N \l__stex_refs_pre_tl
821 \tl_clear:N \l__stex_refs_post_tl
822 \str_clear:N \l__stex_refs_repo_str
823 \keys_set:nn { stex / sref } { #1 }
824 }
825
826 \NewDocumentCommand \sref { 0{} m}{
827 \__stex_refs_args:n { #1 }
828 \str_if_empty:NTF \l__stex_refs_indocument_str {
829 \str_set:Nn \l_tmpa_str { #2 }
830 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
831 \tl_set:Nn \l_tmpa_tl {
832 \l__stex_refs_fallback_tl
833 }
834 \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
835 \str_set:Nn \l_tmpb_str { ##1 }
836 \str_if_eq:eeT { \l_tmpa_str } {
837 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
838 } {

```

```

839     \seq_map_break:n {
840         \tl_set:Nn \l_tmpa_tl {
841             % doc uri in \l_tmpb_str
842             \str_set:Nx \l_tmpa_str {sref_url_\l_tmpb_str_type}
843             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
844                 % reference
845                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
846             }{
847                 % URL
848                 \if_bool:N \c__stex_refs_hyperref_bool {
849                     \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
850                 }{
851                     \l__stex_refs_fallback_tl
852                 }
853             }
854         }
855     }
856 }
857 }
858 \l_tmpa_tl
859 }{
860     % TODO
861 }
862 }
863
864 \end{package}

```


Chapter 21

STEX -Modules Implementation

```
865 <*package>
866
867 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
868
869 <@@=stex_modules>
      Warnings and error messages
870 \msg_new:nnn{stex}{error/unknownmodule}{
871   No~module~#1~found
872 }
873 \msg_new:nnn{stex}{error/syntax}{
874   Syntax~error:~#1
875 }
876 \msg_new:nnn{stex}{error/siglanguage}{
877   Module~#1~declares~signature~#2,~but~does~not~
878   declare~its~language
879 }
```

`\l_stex_current_module_prop` The current module:

```
880 \prop_new:N \l_stex_current_module_prop
```

(End definition for `\l_stex_current_module_prop`. This variable is documented on page 15.)

`\l_stex_all_modules_seq` Stores all available modules

```
881 \seq_new:N \l_stex_all_modules_seq
```

(End definition for `\l_stex_all_modules_seq`. This variable is documented on page 15.)

`\g_stex_modules_in_file_seq` All modules sorted by containing file; used e.g. in `\importmodule`
`\g_stex_module_files_prop`

```
882 \seq_new:N \g_stex_modules_in_file_seq
883 \prop_new:N \g_stex_module_files_prop
```

(End definition for `\g_stex_modules_in_file_seq` and `\g_stex_module_files_prop`. These variables are documented on page 16.)

```

\stex_if_in_module_p:
\stex_if_in_module:TF
884 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
885   \prop_if_empty:NTF \l_stex_current_module_prop
886   \prg_return_false: \prg_return_true:
887 }

```

(End definition for \stex_if_in_module:TF. This function is documented on page 16.)

```

\stex_if_module_exists_p:n
\stex_if_module_exists:nTF
888 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
889   \prop_if_exist:cTF { c_stex_module_#1_prop }
890   \prg_return_true: \prg_return_false:
891 }

```

(End definition for \stex_if_module_exists:nTF. This function is documented on page 16.)

```

\stex_add_to_current_module:n
\STEXexport
892 \cs_new_protected:Nn \stex_add_to_current_module:n {
893   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
894   \tl_put_right:Nn \l_tmpa_tl { #1 }
895   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
896 }
897 \cs_new_protected:Npn \STEXexport {
898   \begingroup
899   \newlinechar=-1\relax
900   \endlinechar=-1\relax
901   %\catcode'\ = 9\relax
902   \expandafter\endgroup\STEXexport:n
903 }
904 \cs_new_protected:Nn \STEXexport:n {
905   \ignorespaces #1
906   \stex_add_to_current_module:n { \ignorespaces #1 }
907   \stex_smsmode_set_codes:
908 }
909 \stex_deactivate_macro:Nn \STEXexport {module~environments}

```

(End definition for \stex_add_to_current_module:n and \STEXexport. These functions are documented on page 16.)

```

\stex_add_constant_to_current_module:n
910 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
911   \str_set:Nx \l_tmpa_str { #1 }
912   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
913   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
914   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
915 }

```

(End definition for \stex_add_constant_to_current_module:n. This function is documented on page 16.)

```

\stex_add_import_to_current_module:n
916 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
917   \str_set:Nx \l_tmpa_str { #1 }
918   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
919   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
920   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
921 }

```

(End definition for `\stex_add_import_to_current_module:n`. This function is documented on page 16.)

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

```

922 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
923   \str_set:Nx \l_tmpa_str { #1 }
924   \seq_set_eq:NN \l_tmpa_seq #2
925   % split off file extension
926   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
927   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
928   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
929   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
930
931   \bool_set_true:N \l_tmpa_bool
932   \bool_while_do:Nn \l_tmpa_bool {
933     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
934     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
935       {source} { \bool_set_false:N \l_tmpa_bool }
936     }{}{
937       \seq_if_empty:NT \l_tmpa_seq {
938         \bool_set_false:N \l_tmpa_bool
939       }
940     }
941   }
942
943   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
944   \str_if_empty:NTF \l_stex_modules_subpath_str {
945     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
946   }{
947     \str_set:Nx \l_stex_modules_ns_str {
948       \l_tmpa_str/\l_stex_modules_subpath_str
949     }
950   }
951 }
```

(End definition for `\stex_modules_compute_namespace:nN`. This function is documented on page 16.)

Stores its return values in:

`\l_stex_modules_ns_str`

```

952 \str_new:N \l_stex_modules_ns_str
953 \str_new:N \l_stex_modules_subpath_str
```

(End definition for `\l_stex_modules_ns_str`. This variable is documented on page ??.)

`\stex_modules_current_namespace:` Computes the current namespace based on the current MathHub repository (if existent) and the current file.

```

954 \cs_new_protected:Nn \stex_modules_current_namespace: {
955   \str_clear:N \l_stex_modules_subpath_str
956   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
957     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
958   }{
959     % split off file extension
960     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
961     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

962     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
963     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
964     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
965     \str_set:Nx \l_stex_modules_ns_str {
966       file:/\stex_path_to_string:N \l_tmpa_seq
967     }
968   }
969 }

```

(End definition for `\stex_modules_current_namespace:.` This function is documented on page 16.)

21.1 The module environment

module arguments:

```

970 \keys_define:nn { stex / module } {
971   title      .str_set_x:N = \l_stex_module_title_str ,
972   ns         .str_set_x:N = \l_stex_module_ns_str ,
973   lang       .str_set_x:N = \l_stex_module_lang_str ,
974   sig        .str_set_x:N = \l_stex_module_sig_str ,
975   creators   .str_set_x:N = \l_stex_module_creators_str ,
976   contributors .str_set_x:N = \l_stex_module_contributors_str ,
977   meta       .str_set_x:N = \l_stex_module_meta_str
978 }
979
980 \cs_new_protected:Nn \__stex_modules_args:n {
981   \str_clear:N \l_stex_module_title_str
982   \str_clear:N \l_stex_module_ns_str
983   \str_clear:N \l_stex_module_lang_str
984   \str_clear:N \l_stex_module_sig_str
985   \str_clear:N \l_stex_module_creators_str
986   \str_clear:N \l_stex_module_contributors_str
987   \str_clear:N \l_stex_module_meta_str
988   \keys_set:nn { stex / module } { #1 }
989 }
990
991 % module parameters here? In the body?
992

```

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

```

993 \cs_new_protected:Nn \stex_module_setup:nn {
994   \str_set:Nx \l_stex_module_name_str { #2 }
995   \__stex_modules_args:n { #1 }

```

First, we set up the name and namespace of the module.
Are we in a nested module?

```

996   \stex_if_in_module:TF {
997     % Nested module
998     \prop_get:NnN \l_stex_current_module_prop
999       { ns } \l_stex_module_ns_str
1000     \str_set:Nx \l_stex_module_name_str {
1001       \prop_item:Nn \l_stex_current_module_prop
1002         { name } / \l_stex_module_name_str
1003     }

```

```

1004 }{
1005   % not nested:
1006   \str_if_empty:NT \l_stex_module_ns_str {
1007     \stex_modules_current_namespace:
1008     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1009     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1010       / {\l_stex_module_ns_str}
1011     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1012     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1013       \str_set:Nx \l_stex_module_ns_str {
1014         \stex_path_to_string:N \l_tmpa_seq
1015       }
1016     }
1017   }
1018 }

```

Next, we determine the language of the module:

```

1019 \str_if_empty:NT \l_stex_module_lang_str {
1020   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1021   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1022   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1023   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1024   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1025     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1026       inferred~from~file~name}
1027     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1028   }
1029 }
1030
1031 \str_if_empty:NF \l_stex_module_lang_str {
1032   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1033     \l_tmpa_str {
1034     \ltx@ifpackageloaded{babel}{
1035       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1036     }{}
1037   } {
1038     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
1039   }
1040 }

```

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

```

1041 \str_if_empty:NTF \l_stex_module_sig_str {
1042   \str_clear:N \l_tmpa_str
1043   \seq_clear:N \l_tmpa_seq
1044   \tl_clear:N \l_tmpa_tl
1045   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
1046     name      = \l_stex_module_name_str ,
1047     ns        = \l_stex_module_ns_str ,
1048     imports   = \exp_not:o { \l_tmpa_seq } ,
1049     constants = \exp_not:o { \l_tmpa_seq } ,
1050     content   = \exp_not:o { \l_tmpa_tl } ,
1051     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1052     lang      = \l_stex_module_lang_str ,

```

```

1053     sig      = \l_stex_module_sig_str ,
1054     meta     = \l_stex_module_meta_str
1055   }
1056 }{
1057   \str_if_empty:NT \l_stex_module_lang_str {
1058     \msg_error:nnnn{stex}{error/siglanguage}{
1059       \l_stex_module_ns_str?\l_stex_module_name_str
1060     }\l_stex_module_sig_str}
1061   }
1062
1063   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1064   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1065   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1066   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1067   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1068   \str_set:Nx \l_tmpa_str {
1069     \stex_path_to_string:N \l_tmpa_seq /
1070     \l_tmpa_str . \l_stex_module_sig_str .tex
1071   }
1072   \IfFileExists \l_tmpa_str {
1073     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1074       \seq_clear:N \l_stex_all_modules_seq
1075       \prop_clear:N \l_stex_current_module_prop
1076       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1077       \input { \l_tmpa_str }
1078     }
1079   }{
1080     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1081   }
1082   \stex_activate_module:n {
1083     \l_stex_module_ns_str ? \l_stex_module_name_str
1084   }
1085   \prop_set_eq:Nc \l_stex_current_module_prop {
1086     c_stex_module_
1087     \l_stex_module_ns_str ?
1088     \l_stex_module_name_str
1089     _prop
1090   }
1091 }

```

We load the metatheory:

```

1092   \str_if_empty:NT \l_stex_module_meta_str {
1093     \str_set:Nx \l_stex_module_meta_str {
1094       \c_stex_metatheory_ns_str ? Metatheory
1095     }
1096   }
1097   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1098     \exp_args:Nx \stex_add_to_current_module:n {
1099       \stex_activate_module:n {\l_stex_module_meta_str}
1100     }
1101     \stex_activate_module:n {\l_stex_module_meta_str}
1102   }
1103 }

```

(End definition for `\stex_module_setup:nn`. This function is documented on page 17.)

module The module environment.

_stex_modules_begin_module:nn implements \begin{module}

```

1104 \cs_new_protected:Nn \_stex_modules_begin_module:nn {
1105   \stex_reactivate_macro:N \STEXexport
1106   \stex_reactivate_macro:N \importmodule
1107   \stex_reactivate_macro:N \symdecl
1108   \stex_reactivate_macro:N \notation
1109   \stex_reactivate_macro:N \symdef
1110   \stex_module_setup:nn{#1}{#2}
1111
1112   \stex_debug:nn{modules}{
1113     New~module:\\
1114     Namespace:~\l_stex_module_ns_str\\
1115     Name:~\l_stex_module_name_str\\
1116     Language:~\l_stex_module_lang_str\\
1117     Signature:~\l_stex_module_sig_str\\
1118     Metatheory:~\l_stex_module_meta_str\\
1119     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1120   }
1121
1122   \seq_put_right:Nx \l_stex_all_modules_seq {
1123     \l_stex_module_ns_str ? \l_stex_module_name_str
1124   }
1125
1126   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1127     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1128
1129   \stex_if_smsmode:TF {
1130     \stex_smsmode_set_codes:
1131   } {
1132     \begin{stex_annotate_env} {theory} {
1133       \l_stex_module_ns_str ? \l_stex_module_name_str
1134     }
1135
1136     \stex_annotate_invisible:nnn{header}{} {
1137       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1138       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1139       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1140         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1141       }
1142     }
1143   }
1144   % TODO: Inherit metatheory for nested modules?
1145 }
1146 \iffalse \end{stex_annotate_env} \fi %^^A make syntax highlighting work again

```

(End definition for _stex_modules_begin_module:nn.)

_stex_modules_end_module: implements \end{module}

```

1147 \cs_new_protected:Nn \_stex_modules_end_module: {
1148   \str_set:Nx \l_tmpa_str {
1149     c_stex_module_
1150     \prop_item:Nn \l_stex_current_module_prop { ns } ?

```

```

1151     \prop_item:Nn \l_stex_current_module_prop { name }
1152     _prop
1153   }
1154   %^^A \prop_new:c { \l_tmpa_str }
1155   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1156   \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1157 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1158 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1159 \NewDocumentEnvironment { @module } { 0 } { m } {
1160   \par
1161   \_stex_modules_begin_module:nn{#1}{#2}
1162 } {
1163   \_stex_modules_end_module:
1164   \stex_if_smsmode:TF {
1165     \exp_args:Nx \stex_add_to_sms:n {
1166       \prop_gset_from_keyval:cn {
1167         c_stex_module_
1168         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1169         \prop_item:Nn \l_stex_current_module_prop { name }
1170         _prop
1171       } {
1172         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1173         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1174         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1175         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1176         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1177         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1178         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1179         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1180         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1181       }
1182     }
1183   }{
1184     \end{stex_annotate_env}
1185   }
1186 }

```

\stex_modules_heading: Code for document headers

```

1187 \cs_if_exist:NTF \thesection {
1188   \newcounter{module}[section]
1189 }{
1190   \newcounter{module}
1191 }
1192
1193 \bool_if:NT \c_stex_showmods_bool {
1194   \latexml_if:F { \RequirePackage{mdframed} }
1195 }
1196
1197 \cs_new_protected:Nn \stex_modules_heading: {
1198   \stepcounter{module}

```



```

1199 \par
1200 \bool_if:NT \c_stex_showmods_bool {
1201   \noindent{\textbf{Module}} ~
1202   \cs_if_exist:NT \thesection {\thesection.}
1203   \themodule ~ [\l_stex_module_name_str]
1204 }
1205 \str_if_empty:NTF \l_stex_module_title_str {
1206 }{
1207   \quad(\l_stex_module_title_str)\hfill
1208 }\par
1209 }
1210 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1211 % TODO
1212 \stex_ref_new_doc_target:n \l_stex_module_name_str
1213 }

```

(End definition for `\stex_modules_heading:`. This function is documented on page 17.)

Finally:

```

1214 \NewDocumentEnvironment { module } { 0 } { m } {
1215   \bool_if:NT \c_stex_showmods_bool {
1216     \begin{mdframed}
1217   }
1218   \begin{@module}[#1]{#2}
1219   \stex_modules_heading:
1220 }{
1221   \end{@module}
1222   \bool_if:NT \c_stex_showmods_bool {
1223     \end{mdframed}
1224   }
1225 }

```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1226 \NewDocumentCommand \STEXModule { m } {
1227   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1228   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1229   \tl_set:Nn \l_tmpa_tl {
1230     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1231   }
1232   \seq_map_inline:Nn \l_stex_all_modules_seq {
1233     \str_set:Nn \l_tmpb_str { ##1 }
1234     \str_if_eq:eeT { \l_tmpa_str } {
1235       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1236     } {
1237       \seq_map_break:n {
1238         \tl_set:Nn \l_tmpa_tl {
1239           \stex_invoke_module:n { ##1 }
1240         }
1241       }
1242     }
1243   }
1244   \l_tmpa_tl

```

```

1245 }
1246
1247 \cs_new_protected:Nn \stex_invoke_module:n {
1248   \stex_debug:nn{modules}{Invoking~module~#1}
1249   \peek_charcode_remove:NTF ! {
1250     \__stex_modules_invoke_uri:nN { #1 }
1251   } {
1252     \peek_charcode_remove:NTF ? {
1253       \__stex_modules_invoke_symbol:nn { #1 }
1254     } {
1255       \msg_error:nnn{stex}{error/syntax}{
1256         ?~or~!~expected~after~
1257         \c_backslash_str STEXModule{#1}
1258       }
1259     }
1260   }
1261 }
1262
1263 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1264   \str_set:Nn #2 { #1 }
1265 }
1266
1267 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1268   \stex_invoke_symbol:n{#1?#2}
1269 }

```

(End definition for `\STEXModule` and `\stex_invoke_module:n`. These functions are documented on page 18.)

`\stex_activate_module:n`

```

1270 \cs_new_protected:Nn \stex_activate_module:n {
1271   \stex_debug:nn{modules}{Activating~module~#1}
1272   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1273     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1274     \prop_item:cn { c_stex_module_#1_prop } { content }
1275   }
1276 }

```

(End definition for `\stex_activate_module:n`. This function is documented on page 19.)

```

1277 </package>

```

Chapter 22

sTEX

-Module Inheritance

Implementation

```
1278 <*package>
1279
1280 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1281
```

22.1 SMS Mode

```
1282 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1283 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1284 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1285 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1286
1287 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1288   \makeatletter
1289   \makeatother
1290   \ExplSyntaxOn
1291   \ExplSyntaxOff
1292 }
1293
1294 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1295   \symdef
1296   \importmodule
1297   \notation
1298   \symdecl
1299   \STEXexport
1300 }
1301
1302 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1303   \tl_to_str:n {
1304     module,
1305     @module
```

```

1306 }
1307 }

```

(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 20.)

```

\stex_if_smsmode_p:
\stex_if_smsmode:TF

```

```

1308 \bool_new:N \g__stex_smsmode_bool
1309 \bool_set_false:N \g__stex_smsmode_bool
1310 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1311   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1312 }

```

(End definition for `\stex_if_smsmode:TF`. This function is documented on page 20.)

```

\__stex_smsmode_if_catcodes_p:

```

Checks whether the SMS mode category code scheme is active.

```

\__stex_smsmode_if_catcodes:TF

```

```

1313 \bool_new:N \g__stex_smsmode_catcode_bool
1314 \bool_set_false:N \g__stex_smsmode_catcode_bool
1315 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1316   \bool_if:NTF \g__stex_smsmode_catcode_bool
1317   \prg_return_true: \prg_return_false:
1318 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1319 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1320   \stex_if_smsmode:T {
1321     \__stex_smsmode_if_catcodes:F {
1322       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1323       \exp_after:wN \char_gset_active_eq:NN
1324       \c_backslash_str \__stex_smsmode_cs:
1325       \tex_global:D \char_set_catcode_active:N \
1326       \tex_global:D \char_set_catcode_other:N $
1327       \tex_global:D \char_set_catcode_other:N ^
1328       \tex_global:D \char_set_catcode_other:N _
1329       \tex_global:D \char_set_catcode_other:N &
1330       \tex_global:D \char_set_catcode_other:N ##
1331     }
1332   }
1333 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `\stex_smsmode_set_codes:.` This function is documented on page 20.)

```

\__stex_smsmode_unset_codes:

```

Sets category code scheme back from the one used in SMS mode.

```

1334 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1335   \__stex_smsmode_if_catcodes:T {
1336     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1337     \exp_after:wN \tex_global:D \exp_after:wN
1338     \char_set_catcode_escape:N \c_backslash_str
1339     \tex_global:D \char_set_catcode_math_toggle:N $
1340     \tex_global:D \char_set_catcode_math_superscript:N ^
1341     \tex_global:D \char_set_catcode_math_subscript:N _
1342     \tex_global:D \char_set_catcode_alignment:N &
1343     \tex_global:D \char_set_catcode_parameter:N ##
1344   }
1345 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1346 \cs_new_protected:Nn \stex_in_smsmode:nn {
1347   \vbox_set:Nn \l_tmpa_box {
1348     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1349     \bool_gset_true:N \g__stex_smsmode_bool
1350     \stex_smsmode_set_codes:
1351     #2
1352     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1353     \stex_if_smsmode:F {
1354       \__stex_smsmode_unset_codes:
1355     }
1356   }
1357   \box_clear:N \l_tmpa_box
1358 }

```

(End definition for `\stex_in_smsmode:nn`. This function is documented on page 21.)

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

```

1359 \cs_new_protected:Nn \_stex_smsmode_cs: {
1360   \str_clear:N \l_tmpa_str
1361   \peek_analysis_map_inline:n {
1362     % #1: token (one expansion)
1363     % #2: charcode
1364     % #3 catcode
1365     \token_if_eq_charcode:NNTF ##3 B {
1366       % token is a letter
1367       \exp_args:NN \str_put_right:Nn \l_tmpa_str { ##1 }
1368     } {
1369       \str_if_empty:NTF \l_tmpa_str {
1370         % we don't allow (or need) single non-letter CSs
1371         % for now
1372         \peek_analysis_map_break:
1373       }{
1374         \str_if_eq:onTF \l_tmpa_str { begin } {
1375           \peek_analysis_map_break:n {
1376             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1377           }
1378         } {
1379           \str_if_eq:onTF \l_tmpa_str { end } {
1380             \peek_analysis_map_break:n {
1381               \exp_after:wN \_stex_smsmode_checkend:n ##1
1382             }
1383           } {
1384             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1385             \exp_args:NNNo \exp_args:NN \tl_if_in:NnTF
1386             \g_stex_smsmode_allowedmacros_tl
1387             { \use:c{\l_tmpa_str} } {
1388               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1389               \peek_analysis_map_break:n {
1390                 \exp_after:wN \l_tmpa_tl ##1
1391               }

```

```

1392     } {
1393         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1394         \g_stex_smsmode_allowedmacros_escape_tl
1395         { \use:c{\l_tmpa_str} } {
1396             \__stex_smsmode_unset_codes:
1397             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1398             % TODO \__stex_smsmode_rescan_cs:
1399             % \int_compare:nNnTF {##2} = {92} {
1400             %     \peek_analysis_map_break:n {
1401             %         \__stex_smsmode_unset_codes:
1402             %         \__stex_smsmode_rescan_cs:
1403             %     }
1404             % } {
1405             %     \peek_analysis_map_break:n {
1406             %         \exp_after:wN \l_tmpa_tl ##1
1407             %     }
1408             % }
1409             } {
1410                 \int_compare:nNnTF {##2} = {92} {
1411                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1412                 }{
1413                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1414                 }
1415             }
1416         }
1417     }
1418 }
1419 }
1420 }
1421 }
1422 }

```

(End definition for __stex_smsmode_cs:.)

__stex_smsmode_rescan_cs: If the last token gobbled by \stex_smsmode_cs: happened to be a \, we need to rescan the cs name and reinsert it into the input stream:

```

1423 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1424     \str_clear:N \l_tmpb_str
1425     \peek_analysis_map_inline:n {
1426         \token_if_eq_charcode:NNTF ##3 B {
1427             % token is a letter
1428             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1429         } {
1430             \peek_analysis_map_break:n {
1431                 \exp_after:wN \use:c \exp_after:wN {
1432                     \exp_after:wN \l_tmpa_str\exp_after:wN
1433                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1434             }
1435         }
1436     }
1437 }

```

(End definition for __stex_smsmode_rescan_cs:.)

`__stex_smsmode_checkbegin:n` called on `\begin`; checks whether the environment being opened is allowed in SMS mode.

```

1438 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1439   \str_set:Nn \l_tmpa_str { #1 }
1440   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1441     \__stex_smsmode_unset_codes:
1442     \begin{#1}
1443   }
1444 }
```

(End definition for `__stex_smsmode_checkbegin:n`.)

`__stex_smsmode_checkend:n` called on `\end`; checks whether the environment being opened is allowed in SMS mode.

```

1445 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1446   \str_set:Nn \l_tmpa_str { #1 }
1447   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1448     \end{#1}
1449   }
1450 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

1451 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1452 \cs_new_protected:Nn \stex_import_module_uri:nn {
1453   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1454   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1455
1456   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1457   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1458   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1459
1460   \stex_modules_current_namespace:
1461   \bool_lazy_all:nTF {
1462     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1463     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1464     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1465   }{
1466     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1467     \str_set_eq:NN \l_stex_module_ns
1468   }{
1469     \str_if_empty:NT \l__stex_importmodule_archive_str {
1470       \prop_if_empty:NF \l_stex_current_repository_prop {
1471         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1472       }
1473     }
1474     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1475       \str_if_empty:NF \l__stex_importmodule_path_str {
1476         \str_set:Nx \l_stex_module_ns_str {
1477           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1478         }
1479       }
1480     }
```

```

1480   }{
1481     \stex_require_repository:n \l__stex_importmodule_archive_str
1482     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1483       \l_stex_module_ns_str
1484     \str_if_empty:NF \l__stex_importmodule_path_str {
1485       \str_set:Nx \l_stex_module_ns_str {
1486         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1487       }
1488     }
1489   }
1490 }
1491 }

```

(End definition for `\stex_import_module_uri:nn`. This function is documented on page 23.)

<code>\l__stex_importmodule_name_str</code>	Store the return values of <code>\stex_import_module_uri:nn</code> .
<code>\l__stex_importmodule_archive_str</code>	1492 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1493 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1494 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1495 <code>\str_new:N \g__stex_importmodule_file_str</code>

(End definition for `\l__stex_importmodule_name_str` and others.)

```

\stex_import_require_module:nnnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1496 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1497   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1498
1499     % archive
1500     \str_set:Nx \l_tmpa_str { #2 }
1501     \str_if_empty:NTF \l_tmpa_str {
1502       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1503     } {
1504       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1505       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1506       \seq_put_right:Nn \l_tmpa_seq { source }
1507     }
1508
1509     % path
1510     \str_set:Nx \l_tmpb_str { #3 }
1511     \str_if_empty:NTF \l_tmpb_str {
1512       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1513
1514       \ltx@ifpackageloaded{babel} {
1515         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1516           { \language } \l_tmpb_str {
1517           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1518         }
1519       } {
1520         \str_clear:N \l_tmpb_str
1521       }
1522
1523       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1524       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1525         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1526     }{
1527       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1528       \IfFileExists{ \l_tmpa_str.tex }{
1529         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1530       }{
1531         % try english as default
1532         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1533         \IfFileExists{ \l_tmpa_str.en.tex }{
1534           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1535         }{
1536           \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1537         }
1538       }
1539     }
1540
1541   } {
1542     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1543     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1544
1545     \ltx@ifpackageloaded{babel} {
1546       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1547         { \language } \l_tmpb_str {
1548           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1549         }
1550     } {
1551       \str_clear:N \l_tmpb_str
1552     }
1553
1554     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1555
1556     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1557     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1558       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1559     }{
1560       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1561       \IfFileExists{ \l_tmpa_str/#4.tex }{
1562         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1563       }{
1564         % try english as default
1565         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1566         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1567           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1568         }{
1569           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1570           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1571             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1572           }{
1573             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1574             \IfFileExists{ \l_tmpa_str.tex }{
1575               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1576             }{
1577               % try english as default
1578               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1579               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1580         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1581     }{
1582         \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1583     }
1584 }
1585 }
1586 }
1587 }
1588 }
1589 }
1590
1591 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1592 \seq_clear:N \g_stex_modules_in_file_seq
1593 % \exp_args:Nnx \use:nn {
1594     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1595         \seq_clear:N \l_stex_all_modules_seq
1596         \prop_clear:N \l_stex_current_module_prop
1597         \str_set:Nx \l_tmpb_str { #2 }
1598         \str_if_empty:NF \l_tmpb_str {
1599             \stex_set_current_repository:n { #2 }
1600         }
1601         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1602         \input { \g__stex_importmodule_file_str }
1603     }
1604 % }{
1605
1606 % }
1607 \prop_gput:Noo \g_stex_module_files_prop
1608 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1609 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1610
1611 \stex_if_module_exists:nF { #1 ? #4 } {
1612     \msg_error:nnn{stex}{error/unknownmodule}{
1613         #1?#4~(in~file~\g__stex_importmodule_file_str)
1614     }
1615 }
1616 }
1617 \stex_activate_module:n { #1 ? #4 }
1618 }

```

(End definition for `\stex_import_require_module:nnnn`. This function is documented on page 23.)

`\importmodule`

```

1619 \NewDocumentCommand \importmodule { 0{} m } {
1620     \stex_import_module_uri:nn { #1 } { #2 }
1621     \stex_debug:nn{modules}{Importing~module:~
1622         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1623     }
1624     \stex_if_smsmode:F {
1625         \stex_import_require_module:nnnn
1626         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1627         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1628         \stex_annotate_invisible:nnn
1629         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1630 }
1631 \exp_args:Nx \stex_add_to_current_module:n {
1632   \stex_import_require_module:nnnn
1633   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1634   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1635 }
1636 \exp_args:Nx \stex_add_import_to_current_module:n {
1637   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1638 }
1639 \stex_smsmode_set_codes:
1640 }
1641 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

(End definition for `\importmodule`. This function is documented on page [21](#).)

`\usemodule`

```

1642 \NewDocumentCommand \usemodule { 0{} m } {
1643   \stex_if_smsmode:F {
1644     \stex_import_module_uri:nn { #1 } { #2 }
1645     \stex_import_require_module:nnnn
1646     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1647     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1648     \stex_annotate_invisible:nnn
1649     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1650   }
1651   \stex_smsmode_set_codes:
1652 }

```

(End definition for `\usemodule`. This function is documented on page [22](#).)

```

1653 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1654 <*package>
1655
1656 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1657
Warnings and error messages
1658
```

23.1 Symbol Declarations

```
1659 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1660 \seq_new:N \l_stex_all_symbols_seq

(End definition for \l_stex_all_symbols_seq. This variable is documented on page 25.)

\STEXsymbol
1661 \NewDocumentCommand \STEXsymbol { m } {
1662   \stex_get_symbol:n { #1 }
1663   \exp_args:No
1664   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1665 }

(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:
1666 \keys_define:nn { stex / symdecl } {
1667   name      .str_set:N = \l_stex_symdecl_name_str ,
1668   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1669   args      .str_set:N = \l_stex_symdecl_args_str ,
1670   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1671   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1672   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1673   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1674   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1675 }
```

```

1676
1677 \bool_new:N \l_stex_symdecl_make_macro_bool
1678
1679 \cs_new_protected:Nn \__stex_symdecl_args:n {
1680   \str_clear:N \l_stex_symdecl_name_str
1681   \str_clear:N \l_stex_symdecl_args_str
1682   \bool_set_false:N \l_stex_symdecl_local_bool
1683   \tl_clear:N \l_stex_symdecl_type_tl
1684   \tl_clear:N \l_stex_symdecl_definiens_tl
1685
1686   \keys_set:nn { stex / symdecl } { #1 }
1687 }

```

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

```

1688
1689 \NewDocumentCommand \symdecl { s O{} m } {
1690   \__stex_symdecl_args:n { #2 }
1691   \IfBooleanTF #1 {
1692     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1693   } {
1694     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1695   }
1696   \stex_symdecl_do:n { #3 }
1697   \stex_smsmode_set_codes:
1698 }
1699 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

(End definition for `\symdecl`. This function is documented on page 24.)

\stex_symdecl_do:n

```

1700 \cs_new_protected:Nn \stex_symdecl_do:n {
1701   \stex_if_in_module:F {
1702     % TODO throw error? some default namespace?
1703   }
1704
1705   \str_if_empty:NT \l_stex_symdecl_name_str {
1706     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1707   }
1708
1709   \prop_if_exist:cT { g_stex_symdecl_
1710     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1711     \prop_item:Nn \l_stex_current_module_prop {name} ?
1712     \l_stex_symdecl_name_str
1713     _prop
1714   }{
1715     % TODO throw error (beware of circular dependencies)
1716   }
1717
1718   \prop_clear:N \l_tmpa_prop
1719   \prop_put:Nnx \l_tmpa_prop { module } {
1720     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1721     \prop_item:Nn \l_stex_current_module_prop {name}
1722   }

```

```

1723 \seq_clear:N \l_tmpa_seq
1724 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1725 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1726 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1727 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1728
1729 \exp_args:No \stex_add_constant_to_current_module:n {
1730   \l_stex_symdecl_name_str
1731 }
1732
1733 % arity/args
1734 \int_zero:N \l_tmpb_int
1735
1736 \bool_set_true:N \l_tmpa_bool
1737 \str_map_inline:Nn \l_stex_symdecl_args_str {
1738   \token_case_meaning:NnF ##1 {
1739     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1740     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1741     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1742     {\tl_to_str:n a} {
1743       \bool_set_false:N \l_tmpa_bool
1744       \int_incr:N \l_tmpb_int
1745     }
1746     {\tl_to_str:n B} {
1747       \bool_set_false:N \l_tmpa_bool
1748       \int_incr:N \l_tmpb_int
1749     }
1750   }{
1751     \msg_set:nnn{stex}{error/wrongargs}{
1752       args~value~in~symbol~declaration~for~
1753       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1754       \prop_item:Nn \l_stex_current_module_prop {name} ?
1755       \l_stex_symdecl_name_str ~
1756       needs~to~be~
1757       i,~a,~b~or~B,~but~##1~given
1758     }
1759     \msg_error:nn{stex}{error/wrongargs}
1760   }
1761 }
1762 \bool_if:NTF \l_tmpa_bool {
1763   % possibly numeric
1764   \str_if_empty:NTF \l_stex_symdecl_args_str {
1765     \prop_put:Nnn \l_tmpa_prop { args } {}
1766     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1767   }{
1768     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1769     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1770     \str_clear:N \l_tmpa_str
1771     \int_step_inline:nn \l_tmpa_int {
1772       \str_put_right:Nn \l_tmpa_str i
1773     }
1774     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1775   }
1776 } {

```

```

1777 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1778 \prop_put:Nnx \l_tmpa_prop { arity }
1779 { \str_count:N \l_stex_symdecl_args_str }
1780 }
1781 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1782
1783
1784 % semantic macro
1785
1786 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1787   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1788     \prop_item:Nn \l_tmpa_prop { module } ?
1789     \prop_item:Nn \l_tmpa_prop { name }
1790   } }
1791
1792   \bool_if:NF \l_stex_symdecl_local_bool {
1793     \exp_args:Nx \stex_add_to_current_module:n {
1794       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1795         \prop_item:Nn \l_tmpa_prop { module } ?
1796         \prop_item:Nn \l_tmpa_prop { name }
1797       } }
1798     }
1799   }
1800 }
1801
1802 % add to all symbols
1803
1804 \bool_if:NF \l_stex_symdecl_local_bool {
1805   \exp_args:Nx \stex_add_to_current_module:n {
1806     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1807       \prop_item:Nn \l_tmpa_prop { module } ?
1808       \prop_item:Nn \l_tmpa_prop { name }
1809     }
1810   }
1811 }
1812
1813 \stex_debug:nn{symbols}{New~symbol:~
1814   \prop_item:Nn \l_tmpa_prop { module } ?
1815   \prop_item:Nn \l_tmpa_prop { name } ^^J
1816   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1817   Args:~\prop_item:Nn \l_tmpa_prop { args }
1818 }
1819
1820 % circular dependencies require this:
1821
1822 \prop_if_exist:cF {
1823   g_stex_symdecl_
1824   \prop_item:Nn \l_tmpa_prop { module } ?
1825   \prop_item:Nn \l_tmpa_prop { name }
1826   _prop
1827 } {
1828   \prop_gset_eq:cN {
1829     g_stex_symdecl_
1830     \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1831     \prop_item:Nn \l_tmpa_prop { name }
1832     _prop
1833   } \l_tmpa_prop
1834 }
1835
1836 \stex_if_smsmode:TF {
1837   \bool_if:NF \l_stex_symdecl_local_bool {
1838     \exp_args:Nx \stex_add_to_sms:n {
1839       \prop_gset_from_keyval:cn {
1840         g_stex_symdecl_
1841         \prop_item:Nn \l_tmpa_prop { module } ?
1842         \prop_item:Nn \l_tmpa_prop { name }
1843         _prop
1844       } {
1845         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1846         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1847         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1848         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1849         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1850         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1851         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1852         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1853       }
1854       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1855         \prop_item:Nn \l_tmpa_prop { module } ?
1856         \prop_item:Nn \l_tmpa_prop { name }
1857       }
1858     }
1859   }
1860 }{
1861   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1862     \prop_item:Nn \l_tmpa_prop { module } ?
1863     \prop_item:Nn \l_tmpa_prop { name }
1864   }
1865   \stex_if_do_html:T {
1866     \stex_annotate_invisible:nnn {symdecl} {
1867       \prop_item:Nn \l_tmpa_prop { module } ?
1868       \prop_item:Nn \l_tmpa_prop { name }
1869     } {
1870       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1871       \stex_annotate_invisible:nnn{args}{}{
1872         \prop_item:Nn \l_tmpa_prop { args }
1873       }
1874       \stex_annotate_invisible:nnn{macroname}{}{#1}
1875       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1876         \stex_annotate_invisible:nnn{definiens}{}
1877         {\l_stex_symdecl_definiens_tl$}
1878       }
1879     }
1880   }
1881 }
1882 }

```

(End definition for `\stex_symdecl_do:n`. This function is documented on page 25.)

`\stex_get_symbol:n`

```
1883 \str_new:N \l_stex_get_symbol_uri_str
1884
1885 \cs_new_protected:Nn \stex_get_symbol:n {
1886   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1887     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1888   }{
1889     % argument is a string
1890     % is it a command name?
1891     \cs_if_exist:cTF { #1 }{
1892       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1893       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1894       \str_if_empty:NNTF \l_tmpa_str {
1895         \exp_args:Nx \cs_if_eq:NNTF {
1896           \tl_head:N \l_tmpa_tl
1897         } \stex_invoke_symbol:n {
1898           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1899         }{
1900           \__stex_symdecl_get_symbol_from_string:n { #1 }
1901         }
1902       } {
1903         \__stex_symdecl_get_symbol_from_string:n { #1 }
1904       }
1905     }{
1906       % argument is not a command name
1907       \__stex_symdecl_get_symbol_from_string:n { #1 }
1908       % \l_stex_all_symbols_seq
1909     }
1910   }
1911 }
1912
1913 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1914   \str_set:Nn \l_tmpa_str { #1 }
1915   \bool_set_false:N \l_tmpa_bool
1916   \stex_if_in_module:T {
1917     \prop_get:NnN \l_stex_current_module_prop
1918     { constants } \l_tmpa_seq
1919     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1920       \bool_set_true:N \l_tmpa_bool
1921       \str_set:Nx \l_stex_get_symbol_uri_str {
1922         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1923         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1924       }
1925     }
1926   }
1927   \bool_if:NF \l_tmpa_bool {
1928     \tl_set:Nn \l_tmpa_tl {
1929       \msg_set:nnn{stex}{error/unknownsymbol}{
1930         No~symbol~#1~found!
1931       }
1932     }
1933     \msg_error:nn{stex}{error/unknownsymbol}
1934   }
1935   \str_set:Nn \l_tmpa_str { #1 }
1936   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1936 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1937   \str_set:Nn \l_tmpb_str { ##1 }
1938   \str_if_eq:eeT { \l_tmpa_str } {
1939     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1940   } {
1941     \seq_map_break:n {
1942       \tl_set:Nn \l_tmpa_tl {
1943         \str_set:Nn \l_stex_get_symbol_uri_str {
1944           ##1
1945         }
1946       }
1947     }
1948   }
1949 }
1950 \l_tmpa_tl
1951 }
1952 }
1953
1954 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1955   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1956   { \tl_tail:N \l_tmpa_tl }
1957   \tl_if_single:NTF \l_tmpa_tl {
1958     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1959       \exp_after:wN \str_set:Nn \exp_after:wN
1960       \l_stex_get_symbol_uri_str \l_tmpa_tl
1961     }{
1962       % TODO
1963       % tail is not a single group
1964     }
1965   }{
1966     % TODO
1967     % tail is not a single group
1968   }
1969 }

```

(End definition for `\stex_get_symbol:n`. This function is documented on page [25](#).)

23.2 Notations

```

1970 <@@=stex_notation>
1971 notation arguments:
1972 \keys_define:nn { stex / notation } {
1973   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1974   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1975   prec .str_set_x:N = \l__stex_notation_prec_str ,
1976   op .tl_set:N = \l__stex_notation_op_tl ,
1977   unknown .code:n = \str_set:Nx
1978     \l__stex_notation_variant_str \l_keys_key_str
1979 }
1980 \cs_new_protected:Nn \__stex_notation_args:n {
1981   \str_clear:N \l__stex_notation_lang_str
1982   \str_clear:N \l__stex_notation_variant_str

```

```

1983 \str_clear:N \l__stex_notation_prec_str
1984 \tl_clear:N \l__stex_notation_op_tl
1985
1986 \keys_set:nn { stex / notation } { #1 }
1987 }

```

\notation

```

1988 \NewDocumentCommand \notation { 0{ } m } {
1989   \__stex_notation_args:n { #1 }
1990   \tl_clear:N \l_stex_symdecl_definiens_tl
1991   \stex_get_symbol:n { #2 }
1992   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1993 }
1994 \stex_deactivate_macro:Nn \notation {module~environments}

```

(End definition for \notation. This function is documented on page 25.)

\stex_notation_do:nn

```

1995 \cs_new_protected:Nn \stex_notation_do:nn {
1996   \prop_set_eq:Nc \l_tmpa_prop {
1997     g_stex_symdecl_ #1 _prop
1998   }
1999
2000   \prop_clear:N \l_tmpb_prop
2001   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2002   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2003   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2004
2005   % precedences
2006   \seq_clear:N \l_tmpb_seq
2007   \exp_args:NNno
2008   \str_if_empty:NTF \l__stex_notation_prec_str {
2009     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2010     \int_compare:nNnTF \l_tmpa_str = 0 {
2011       \exp_args:NNnx
2012       \prop_put:Nno \l_tmpb_prop { opprec }
2013       { \neginfprec }
2014     }{
2015       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2016     }
2017   } {
2018     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2019       \exp_args:NNnx
2020       \prop_put:Nno \l_tmpb_prop { opprec }
2021       { \neginfprec }
2022       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2023       \int_step_inline:nn { \l_tmpa_str } {
2024         \exp_args:NNx
2025         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2026       }
2027     }{
2028       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2029       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2030         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2031         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2032         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2033         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2034         \seq_map_inline:Nn \l_tmpa_seq {
2035             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2036         }
2037     }
2038     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2039 }{
2040     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2041     \int_compare:nNnTF \l_tmpa_str = 0 {
2042         \exp_args:NNnx
2043         \prop_put:Nno \l_tmpb_prop { opprec }
2044         { \infprec }
2045     }{
2046         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2047     }
2048 }
2049 }
2050 }
2051
2052 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2053 \int_step_inline:nn { \l_tmpa_str } {
2054     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2055         \exp_args:NNx
2056         \seq_put_right:Nn \l_tmpb_seq {
2057             \prop_item:Nn \l_tmpb_prop { opprec }
2058         }
2059     }
2060 }
2061
2062 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2063 \tl_clear:N \l_tmpa_tl
2064
2065 \int_compare:nNnTF \l_tmpa_str = 0 {
2066     \exp_args:NNe
2067     \cs_set:Npn \l__stex_notation_macrocode_cs {
2068         \_stex_term_math_oms:nnnn { #1 }
2069         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2070         { \prop_item:Nn \l_tmpb_prop { opprec } }
2071         { \exp_not:n { #2 } }
2072     }
2073     \__stex_notation_final:
2074 }{
2075     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2076     \str_if_in:NnTF \l_tmpb_str b {
2077         \exp_args:Nne \use:nn
2078         {
2079             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2080             \cs_set:Npn \l_tmpa_str { {
2081                 \_stex_term_math_omb:nnnn { #1 }
2082                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2083                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2084                 { \exp_not:n { #2 } }
2085             }}

```

```

2086   }{
2087     \str_if_in:NnTF \l_tmpb_str B {
2088       \exp_args:Nne \use:nn
2089       {
2090         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2091         \cs_set:Npn \l_tmpa_str } { {
2092           \stex_term_math_omb:nnnn { #1 }
2093           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2094           { \prop_item:Nn \l_tmpb_prop { opprec } }
2095           { \exp_not:n { #2 } }
2096         } }
2097       }{
2098         \exp_args:Nne \use:nn
2099         {
2100           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2101           \cs_set:Npn \l_tmpa_str } { {
2102             \stex_term_math_oma:nnnn { #1 }
2103             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2104             { \prop_item:Nn \l_tmpb_prop { opprec } }
2105             { \exp_not:n { #2 } }
2106           } }
2107         }
2108       }
2109
2110       \int_zero:N \l_tmpa_int
2111       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2112       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2113       \__stex_notation_arguments:
2114     }
2115   }

```

(End definition for `\stex_notation_do:nn`. This function is documented on page 26.)

`__stex_notation_arguments:` Takes care of annotating the arguments in a notation macro

```

2116 \cs_new_protected:Nn \__stex_notation_arguments: {
2117   \int_incr:N \l_tmpa_int
2118   \str_if_empty:NnTF \l_tmpa_str {
2119     \__stex_notation_final:
2120   }{
2121     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2122     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2123     \str_if_eq:NnTF \l_tmpb_str a {
2124       \__stex_notation_argument_assoc:n
2125     }{
2126       \str_if_eq:NnTF \l_tmpb_str B {
2127         \__stex_notation_argument_assoc:n
2128       }{
2129         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2130         \tl_put_right:Nx \l_tmpa_tl {
2131           { \stex_term_math_arg:nnn
2132             { \int_use:N \l_tmpa_int }
2133             { \l_tmpb_str }
2134             { ####\int_use:N \l_tmpa_int }
2135           }

```

```

2136     }
2137     \__stex_notation_arguments:
2138   }
2139 }
2140 }
2141 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2142 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2143   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2144   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2145   \tl_put_right:Nx \l_tmpa_tl {
2146     { \_stex_term_math_assoc_arg:nnnn
2147       { \int_use:N \l_tmpa_int }
2148       { \l_tmpb_str }
2149       \exp_args:No \exp_not:n
2150       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2151       { ####\int_use:N \l_tmpa_int }
2152     }
2153   }
2154   \__stex_notation_arguments:
2155 }

```

(End definition for _stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2156 \cs_new_protected:Nn \__stex_notation_final: {
2157   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2158   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2159   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2160   \exp_args:Nne \use:nn
2161   {
2162     \cs_generate_from_arg_count:cNnn {
2163       stex_notation_ \l_tmpa_str \c_hash_str
2164       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2165       _cs
2166     }
2167     \cs_gset:Npn \l_tmpb_str { { {
2168       \exp_after:wN \exp_after:wN \exp_after:wN
2169       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2170       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2171     } } }
2172
2173     \tl_if_empty:NF \l__stex_notation_op_tl {
2174       \cs_gset:cpx {
2175         stex_op_notation_ \l_tmpa_str \c_hash_str
2176         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2177         _cs
2178       } {
2179         \_stex_term_oms:nnn {
2180           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2181           \l__stex_notation_lang_str

```

```

2182     }{
2183         \l_tmpa_str
2184     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2185 }
2186 }
2187
2188
2189
2190 \stex_debug:nn{symbols}{
2191     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2192     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2193     Operator~precedence:~
2194     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2195     Argument~precedences:~
2196     \seq_use:Nn \l_tmpa_seq {,~}^^J
2197     Notation: \cs_meaning:c {
2198         stex_notation_ \l_tmpa_str \c_hash_str
2199         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2200         _cs
2201     }
2202 }
2203
2204 \prop_gset_eq:cN {
2205     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2206     \c_hash_str \l__stex_notation_lang_str _prop
2207 } \l_tmpb_prop
2208
2209 \exp_args:Nx
2210 \stex_add_to_current_module:n {
2211     \prop_get:cnN {
2212         g_stex_symdecl_
2213         \prop_item:Nn \l_tmpb_prop { symbol }
2214         _prop
2215     } { notations } \exp_not:N \l_tmpa_seq
2216     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2217         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2218     }
2219     \prop_put:cno {
2220         g_stex_symdecl_
2221         \prop_item:Nn \l_tmpb_prop { symbol }
2222         _prop
2223     } { notations } \exp_not:N \l_tmpa_seq
2224 }
2225
2226 \stex_if_smsmode:TF {
2227     \stex_smsmode_set_codes:
2228     \exp_args:Nx \stex_add_to_sms:n {
2229         \prop_gset_from_keyval:cn {
2230             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2231             \c_hash_str \l__stex_notation_lang_str _prop
2232         } {
2233             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2234             language = \prop_item:Nn \l_tmpb_prop { language } ,
2235             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2236         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2237         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2238     }
2239 }
2240 }{
2241   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2242   \seq_put_right:Nx \l_tmpa_seq {
2243     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2244   }
2245   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2246   \prop_set_eq:cN {
2247     g_stex_symdecl_ \l_tmpa_str _prop
2248   } \l_tmpa_prop
2249
2250   % HTML annotations
2251   \stex_if_do_html:T {
2252     \stex_annotate_invisible:nnn { notation }
2253     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2254       \stex_annotate_invisible:nnn { notationfragment }
2255       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2256       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2257       \stex_annotate_invisible:nnn { precedence }
2258       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2259         \seq_use:Nn \l_tmpa_seq { x }
2260       }{}
2261
2262       \int_zero:N \l_tmpa_int
2263       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2264       \tl_clear:N \l_tmpa_tl
2265       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2266         \int_incr:N \l_tmpa_int
2267         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2268         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2269         \str_if_eq:VnTF \l_tmpb_str a {
2270           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2271             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2272             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2273           } }
2274         }{
2275           \str_if_eq:VnTF \l_tmpb_str B {
2276             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2277               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2278               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2279             } }
2280           }{
2281             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2282               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2283             } }
2284           }
2285         }
2286       }
2287       \stex_annotate_invisible:nnn { notationcomp }{}{
2288         $ \exp_args:Nno \use:nn { \use:c {
2289           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```



```

2290         \c_hash_str \l__stex_notation_variant_str
2291         \c_hash_str \l__stex_notation_lang_str _cs
2292     } } { \l_tmpa_tl } $
2293   }
2294 }
2295 }
2296 }
2297 }

```

(End definition for _stex_notation_final:.)

\symdef

```

2298 \keys_define:nn { stex / symdef } {
2299   name      .str_set:x:N = \l_stex_symdecl_name_str ,
2300   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2301   args      .str_set:x:N = \l_stex_symdecl_args_str ,
2302   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
2303   def       .tl_set:N    = \l_stex_symdecl_definiens_tl ,
2304   op        .tl_set:N    = \l__stex_notation_op_tl ,
2305   lang      .str_set:x:N = \l__stex_notation_lang_str ,
2306   variant   .str_set:x:N = \l__stex_notation_variant_str ,
2307   prec      .str_set:x:N = \l__stex_notation_prec_str ,
2308   unknown   .code:n      = \str_set:Nx
2309             \l__stex_notation_variant_str \l_keys_key_str
2310 }
2311
2312 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2313   \str_clear:N \l_stex_symdecl_name_str
2314   \str_clear:N \l_stex_symdecl_args_str
2315   \bool_set_false:N \l_stex_symdecl_local_bool
2316   \tl_clear:N \l_stex_symdecl_type_tl
2317   \tl_clear:N \l_stex_symdecl_definiens_tl
2318   \str_clear:N \l__stex_notation_lang_str
2319   \str_clear:N \l__stex_notation_variant_str
2320   \str_clear:N \l__stex_notation_prec_str
2321   \tl_clear:N \l__stex_notation_op_tl
2322
2323   \keys_set:nn { stex / symdef } { #1 }
2324 }
2325
2326 \NewDocumentCommand \symdef { 0{} m } {
2327   \__stex_notation_symdef_args:n { #1 }
2328   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2329   \stex_symdecl_do:n { #2 }
2330   \exp_args:Nx \stex_notation_do:nn {
2331     \prop_item:Nn \l_tmpa_prop { module } ?
2332     \prop_item:Nn \l_tmpa_prop { name }
2333   }
2334 }
2335 \stex_deactivate_macro:Nn \symdef {module~environments}

```

(End definition for \symdef. This function is documented on page 26.)

```

2336 \</package>

```

Chapter 24

STEX -Terms Implementation

```
2337 <*package>
2338
2339 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2340
2341 <@@=stex_terms>
2342
2343   Warnings and error messages
2344   \msg_new:nnn{stex}{error/nonotation}{
2345     Symbol~#1~invoked,~but~has~no~notation~#2!
2346   }
2347   \msg_new:nnn{stex}{error/notationarg}{
2348     Error~in~parsing~notation~#1
2349   }
2350   \msg_new:nnn{stex}{error/noop}{
2351     Symbol~#1~has~no~operator~notation~for~notation~#2
2352   }
```

24.1 Symbol Invocations

Arguments:

```
2352 \keys_define:nn { stex / terms } {
2353   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2354   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2355   unknown .code:n = \str_set:Nx
2356     \l__stex_terms_variant_str \l_keys_key_str
2357 }
2358
2359 \cs_new_protected:Nn \__stex_terms_args:n {
2360   \str_clear:N \l__stex_terms_lang_str
2361   \str_clear:N \l__stex_terms_variant_str
2362   \str_clear:N \l__stex_terms_prec_str
2363   \tl_clear:N \l__stex_terms_op_tl
2364
2365   \keys_set:nn { stex / terms } { #1 }
```

2366 }

\stex_invoke_symbol:n Invokes a semantic macro

```

2367 \cs_new_protected:Nn \stex_invoke_symbol:n {
2368   \if_mode_math:
2369     \exp_after:wN \__stex_terms_invoke_math:n
2370   \else:
2371     \exp_after:wN \__stex_terms_invoke_text:n
2372   \fi: { #1 }
2373 }
```

(End definition for \stex_invoke_symbol:n. This function is documented on page 27.)

__stex_terms_invoke_math:n

```

2374 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2375   \peek_charcode_remove:NTF ! {
2376     \peek_charcode:NTF [ {
2377       \__stex_terms_invoke_op:nw { #1 }
2378     }{
2379       \peek_charcode_remove:NTF ! {
2380         \peek_charcode:NTF [ {
2381           \__stex_terms_invoke_op_custom:nw
2382         }{
2383           % TODO throw error
2384         }
2385       }{
2386         \__stex_terms_invoke_op:nw { #1 } []
2387       }
2388     }{
2389       \peek_charcode_remove:NTF * {
2390         \__stex_terms_invoke_text:n { #1 }
2391       }{
2392         \peek_charcode:NTF [ {
2393           \__stex_terms_invoke_math:nw { #1 }
2394         }{
2395           \__stex_terms_invoke_math:nw { #1 } []
2396         }
2397       }
2398     }
2399   }
2400 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```

2401 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2402   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2403     \stex_highlight_term:nn{#1}{#2}
2404   }
2405 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2406 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2407   \_stex_terms_args:n { #2 }
2408   \cs_if_exist:cTF {
2409     stex_op_notation_ #1 \c_hash_str
2410     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2411   }{
2412     \csname stex_op_notation_ #1 \c_hash_str
2413       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2414     \endcsname
2415   }{
2416     \msg_error:nnnn{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str}
2417   }
2418 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2419 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2420   \_stex_terms_args:n { #2 }
2421   \prop_set_eq:Nc \l_tmpa_prop {
2422     g_stex_symdecl_ #1 _prop
2423   }
2424   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2425   \seq_if_empty:NTF \l_tmpa_seq {
2426     \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2427   } {
2428     \seq_if_in:NxTF \l_tmpa_seq
2429     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2430       \use:c{
2431         stex_notation_ #1 \c_hash_str
2432         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2433         _cs
2434       }
2435     }{
2436       \str_if_empty:NTF \l__stex_terms_variant_str {
2437         \str_if_empty:NTF \l__stex_terms_lang_str {
2438           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2439           \use:c{
2440             stex_notation_ #1 \c_hash_str \l_tmpa_str
2441             _cs
2442           }
2443         }{
2444           \msg_error:nn{stex}{error/nonotation}{#1}{
2445             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2446           }
2447         }
2448       }{
2449         \msg_error:nn{stex}{error/nonotation}{#1}{
2450           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2451         }
2452       }
2453     }
2454   }

```

```
2455 }
(End definition for \_stex_terms_invoke_math:nw.)
```

_stex_terms_invoke_text:n

```
2456 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2457   \peek_charcode_remove:NTF ! {
2458     \stex_term_custom:nn { #1 } { }
2459   }{
2460     \prop_set_eq:Nc \l_tmpa_prop {
2461       g_stex_symdecl_ #1 _prop
2462     }
2463     \prop_get:Nn \l_tmpa_prop { args } \l_tmpa_str
2464     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2465   }
2466 }
```

(End definition for _stex_terms_invoke_text:n.)

24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2467 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2468 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2469 \int_new:N \l__stex_terms_downprec
2470 \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 28.)

Bracketing:

```
\l_stex_terms_left_bracket_str
\l_stex_terms_right_bracket_str
2471 \tl_set:Nn \l__stex_terms_left_bracket_str (
2472 \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

```
2473 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2474   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2475     \bool_set_false:N \l__stex_terms_brackets_done_bool
2476     #2
2477   } {
2478     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2479       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2480         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2481         \dobrackets { #2 }
2482       }
2483     }{ #2 }
2484   }
2485 }
```

(End definition for _stex_terms_maybe_brackets:nn.)

`\dobrackets`

```
2486 \bool_new:N \l__stex_terms_brackets_done_bool
2487 %\RequirePackage{scalerel}
2488 \cs_new_protected:Npn \dobrackets #1 {
2489   %\ThisStyle{\if D\m@switch
2490   %    \exp_args:Nnx \use:nn
2491   %    { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2492   %    { \exp_not:N\right\l__stex_terms_right_bracket_str }
2493   % \else
2494   \exp_args:Nnx \use:nn
2495   {
2496     \bool_set_true:N \l__stex_terms_brackets_done_bool
2497     \int_set:Nn \l__stex_terms_downprec \infpref
2498     \l__stex_terms_left_bracket_str
2499     #1
2500   }
2501   {
2502     \bool_set_false:N \l__stex_terms_brackets_done_bool
2503     \l__stex_terms_right_bracket_str
2504     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2505   }
2506   %\fi}
2507 }
```

(End definition for `\dobrackets`. This function is documented on page 28.)

`\withbrackets`

```
2508 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2509   \exp_args:Nnx \use:nn
2510   {
2511     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2512     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2513     #3
2514   }
2515   {
2516     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2517     {\l__stex_terms_left_bracket_str}
2518     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2519     {\l__stex_terms_right_bracket_str}
2520   }
2521 }
```

(End definition for `\withbrackets`. This function is documented on page 28.)

`\STEXinvisible`

```
2522 \cs_new_protected:Npn \STEXinvisible #1 {
2523   \stex_annotate_invisible:n { #1 }
2524 }
```

(End definition for `\STEXinvisible`. This function is documented on page 29.)

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2525 \cs_new_protected:Nn \_stex_term_oms:nnn {
2526   \stex_annotate:nnn{ OMID }{ #2 }{
2527     \stex_highlight_term:nn { #1 } { #3 }
2528   }
2529 }
2530
2531 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2532   \__stex_terms_maybe_brackets:nn { #3 }{
2533     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2534   }
2535 }

```

(End definition for `_stex_term_math_oms:nnnn`. This function is documented on page 27.)

`_stex_term_math_oma:nnnn`

```

2536 \cs_new_protected:Nn \_stex_term_oma:nnn {
2537   \stex_annotate:nnn{ OMA }{ #2 }{
2538     \stex_highlight_term:nn { #1 } { #3 }
2539   }
2540 }
2541
2542 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2543   \__stex_terms_maybe_brackets:nn { #3 }{
2544     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2545   }
2546 }

```

(End definition for `_stex_term_math_oma:nnnn`. This function is documented on page 27.)

`_stex_term_math_omb:nnnn`

```

2547 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2548   \stex_annotate:nnn{ OMBIND }{ #2 }{
2549     \stex_highlight_term:nn { #1 } { #3 }
2550   }
2551 }
2552
2553 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2554   \__stex_terms_maybe_brackets:nn { #3 }{
2555     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2556   }
2557 }

```

(End definition for `_stex_term_math_omb:nnnn`. This function is documented on page 27.)

`_stex_term_math_arg:nnn`

```

2558 \cs_new_protected:Nn \_stex_term_arg:nn {
2559   \stex_unhighlight_term:n {
2560     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2561   }
2562 }
2563 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2564   \exp_args:Nnx \use:nn
2565     { \int_set:Nn \l__stex_terms_downprec { #2 }

```

```

2566     \stex_term_arg:nn { #1 }{ #3 }
2567   }
2568   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2569 }

```

(End definition for `\stex_term_math_arg:nnn`. This function is documented on page 27.)

`\stex_term_math_assoc_arg:nnnn`

```

2570 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2571   \clist_set:Nn \l_tmpa_clist{ #4 }
2572   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2573     \tl_set:Nn \l_tmpa_tl { #4 }
2574   }{
2575     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2576     \clist_reverse:N \l_tmpa_clist
2577     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2578
2579     \clist_map_inline:Nn \l_tmpa_clist {
2580       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2581         \exp_args:Nno
2582         \l_tmpa_cs { ##1 } \l_tmpa_tl
2583       }
2584     }
2585
2586   }
2587   \exp_args:Nnno
2588   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2589 }

```

(End definition for `\stex_term_math_assoc_arg:nnnn`. This function is documented on page 27.)

`\stex_term_custom:nn`

```

2590 \cs_new_protected:Nn \stex_term_custom:nn {
2591   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2592   \str_set:Nn \l_tmpa_str { #2 }
2593   \tl_clear:N \l_tmpa_tl
2594   \int_zero:N \l_tmpa_int
2595   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2596   \__stex_terms_custom_loop:
2597 }

```

(End definition for `\stex_term_custom:nn`. This function is documented on page 29.)

`__stex_terms_custom_loop:`

```

2598 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2599   \bool_set_false:N \l_tmpa_bool
2600   \bool_while_do:nn {
2601     \str_if_eq_p:ee X {
2602       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2603     }
2604   }{
2605     \int_incr:N \l_tmpa_int
2606   }
2607
2608   \peek_charcode:NTF [ {

```



```

2609 % notation/text component
2610 \__stex_terms_custom_component:w
2611 } {
2612 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2613 % all arguments read => finish
2614 \__stex_terms_custom_final:
2615 } {
2616 % arguments missing
2617 \peek_charcode_remove:NTF * {
2618 % invisible, specific argument position or both
2619 \peek_charcode:NTF [ {
2620 % visible specific argument position
2621 \__stex_terms_custom_arg:wn
2622 } {
2623 % invisible
2624 \peek_charcode_remove:NTF * {
2625 % invisible specific argument position
2626 \__stex_terms_custom_arg_inv:wn
2627 } {
2628 % invisible next argument
2629 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2630 }
2631 }
2632 } {
2633 % next normal argument
2634 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2635 }
2636 }
2637 }
2638 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2639 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2640 \bool_set_true:N \l_tmpa_bool
2641 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2642 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2643 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2644 \str_set:Nx \l_tmpb_str {
2645 \str_item:Nn \l_tmpa_str { #1 }
2646 }
2647 \str_case:VnTF \l_tmpb_str {
2648 { X } {
2649 \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2650 }
2651 { i } { \__stex_terms_custom_set_X:n { #1 } }
2652 { b } { \__stex_terms_custom_set_X:n { #1 } }
2653 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2654 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2655 }{}{

```

```

2656 \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2657 }
2658
2659 \bool_if:nTF \l_tmpa_bool {
2660   \tl_put_right:Nx \l_tmpa_tl {
2661     \stex_annotate_invisible:n {
2662       \stex_term_arg:nn { \int_eval:n { #1 } }
2663       \exp_not:n { { #2 } }
2664     }
2665   }
2666 } {
2667   \tl_put_right:Nx \l_tmpa_tl {
2668     \stex_term_arg:nn { \int_eval:n { #1 } }
2669     \exp_not:n { { #2 } }
2670   }
2671 }
2672
2673 \__stex_terms_custom_loop:
2674 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2675 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2676   \str_set:Nx \l_tmpa_str {
2677     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2678     X
2679     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2680   }
2681 }

```

(End definition for __stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2682 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2683   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2684   \__stex_terms_custom_loop:
2685 }

```

(End definition for _stex_terms_custom_component:.)

__stex_terms_custom_final:

```

2686 \cs_new_protected:Nn \__stex_terms_custom_final: {
2687   \int_compare:nNnTF \l_tmpb_int = 0 {
2688     \exp_args:Nnno \stex_term_oms:nnn
2689   } {
2690     \str_if_in:NnTF \l_tmpa_str {b} {
2691       \exp_args:Nnno \stex_term_ombind:nnn
2692     } {
2693       \exp_args:Nnno \stex_term_oma:nnn
2694     }
2695   }
2696   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2697 }

```

(End definition for `_stex_terms_custom_final:`.)

```

\symref
\symname
2698 \NewDocumentCommand \symref { m m }{
2699   \let\compemph_uri_prev:\compemph@uri
2700   \let\compemph@uri\symrefemph@uri
2701   \STEXsymbol{#1}! [#2]
2702   \let\compemph@uri\compemph_uri_prev:
2703 }
2704
2705 \keys_define:nn { stex / symname } {
2706   post      .str_set_x:N   = \l_stex_symname_post_str
2707 }
2708
2709 \cs_new_protected:Nn \stex_symname_args:n {
2710   \str_clear:N \l_stex_symname_post_str
2711   \keys_set:nn { stex / symname } { #1 }
2712 }
2713
2714 \NewDocumentCommand \symname { 0{} m }{
2715   \stex_symname_args:n { #1 }
2716   \stex_get_symbol:n { #2 }
2717   \str_set:Nx \l_tmpa_str {
2718     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2719   }
2720   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2721
2722   \let\compemph_uri_prev:\compemph@uri
2723   \let\compemph@uri\symrefemph@uri
2724   \exp_args:NNx \use:nn
2725   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2726     \l_tmpa_str \l_stex_symname_post_str
2727   ] }
2728   \let\compemph@uri\compemph_uri_prev:
2729 }

```

(End definition for `\symref` and `\symname`. These functions are documented on page 27.)

24.3 Notation Components

2730 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2731
2732 \str_new:N \l__stex_notationcomps_highlight_uri_str
2733 \cs_new_protected:Nn \stex_highlight_term:nn {
2734   \exp_args:Nnx
2735   \use:nn {
2736     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2737     #2
2738   } {
2739     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2740     { \l__stex_notationcomps_highlight_uri_str }
2741   }

```

```

2742 }
2743
2744 \cs_new_protected:Nn \stex_unhighlight_term:n {
2745 % \latexml_if:TF {
2746 % #1
2747 % } {
2748 % \rustex_if:TF {
2749 % #1
2750 % } {
2751 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2752 % }
2753 % }
2754 }

```

(End definition for `\stex_highlight_term:nn`. This function is documented on page 29.)

```

\comp
\compemph@uri 2755 \cs_new_protected:Npn \comp #1 {
\compemph 2756 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2757 \rustex_if:TF {
\defemph@uri 2758 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2759 }{
\symrefemph@uri 2760 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2761 }
2762 }
2763 }
2764
2765 \cs_new_protected:Npn \compemph@uri #1 #2 {
2766 \compemph{ #1 }
2767 }
2768
2769
2770 \cs_new_protected:Npn \compemph #1 {
2771 \textcolor{blue}{#1}
2772 }
2773
2774 \cs_new_protected:Npn \defemph@uri #1 #2 {
2775 \defemph{#1}
2776 }
2777
2778 \cs_new_protected:Npn \defemph #1 {
2779 \textbf{#1}
2780 }
2781
2782 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2783 \symrefemph{#1}
2784 }
2785
2786 \cs_new_protected:Npn \symrefemph #1 {
2787 \textbf{#1}
2788 }

```

(End definition for `\comp` and others. These functions are documented on page 29.)

\ellipses

```
2789 \NewDocumentCommand \ellipses {} { \ldots }
```

(End definition for \ellipses. This function is documented on page 29.)

```
\parray
\prmatrix 2790 \bool_new:N \l_stex_inarray_bool
\parrayline 2791 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2792 \NewDocumentCommand \parray { m m } {
\parraycell 2793 \begingroup
2794 \bool_set_true:N \l_stex_inarray_bool
2795 \begin{array}{#1}
2796 #2
2797 \end{array}
2798 \endgroup
2799 }
2800
2801 \NewDocumentCommand \prmatrix { m } {
2802 \begingroup
2803 \bool_set_true:N \l_stex_inarray_bool
2804 \begin{matrix}
2805 #1
2806 \end{matrix}
2807 \endgroup
2808 }
2809
2810 \def \parrayline #1 #2 {
2811 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2812 }
2813
2814 \def \parraylineh #1 #2 {
2815 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2816 }
2817
2818 \def \parraycell #1 {
2819 #1 \bool_if:NT \l_stex_inarray_bool {&}
2820 }
```

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

```
2821 \end{package}
```

Chapter 25

STEX -Structural Features Implementation

```
2822 <*package>
2823
2824 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2825
2826 <@@=stex_features>
      Warnings and error messages
2827
```

25.1 The feature environment

structural@feature

```
2828
2829 \NewDocumentEnvironment{structural@feature}{ m m m }{
2830   \stex_if_in_module:F {
2831     \msg_set:nnn{stex}{error/nomodule}{
2832       Structural~Feature~has~to~occur~in~a~module:\\
2833       Feature~#2~of~type~#1\\
2834       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2835     }
2836     \msg_error:nn{stex}{error/nomodule}
2837   }
2838
2839   \str_set:Nx \l_stex_module_name_str {
2840     \prop_item:Nn \l_stex_current_module_prop
2841       { name } / #2 - feature
2842   }
2843
2844   \str_set:Nx \l_stex_module_ns_str {
2845     \prop_item:Nn \l_stex_current_module_prop
2846       { ns }
2847   }
2848
```

```

2849
2850 \str_clear:N \l_tmpa_str
2851 \seq_clear:N \l_tmpa_seq
2852 \tl_clear:N \l_tmpa_tl
2853 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2854   origname = #2,
2855   name      = \l_stex_module_name_str ,
2856   ns        = \l_stex_module_ns_str ,
2857   imports   = \exp_not:o { \l_tmpa_seq } ,
2858   constants = \exp_not:o { \l_tmpa_seq } ,
2859   content    = \exp_not:o { \l_tmpa_tl } ,
2860   file       = \exp_not:o { \g_stex_currentfile_seq } ,
2861   lang       = \l_stex_module_lang_str ,
2862   sig        = \l_tmpa_str ,
2863   meta       = \l_tmpa_str ,
2864   feature    = #1 ,
2865 }
2866
2867 \stex_if_smsmode:TF {
2868   \stex_smsmode_set_codes:
2869 } {
2870   \begin{stex_annotate_env}{ feature:#1 }{}
2871   \stex_annotate_invisible:nnn{header}{}{ #3 }
2872 }
2873 }{
2874   \str_set:Nx \l_tmpa_str {
2875     c_stex_feature_
2876     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2877     \prop_item:Nn \l_stex_current_module_prop { name }
2878     _prop
2879   }
2880   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2881   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2882   \stex_if_smsmode:TF {
2883     \exp_args:Nx \stex_add_to_sms:n {
2884       \prop_gset_from_keyval:cn {
2885         c_stex_feature_
2886         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2887         \prop_item:Nn \l_stex_current_module_prop { name }
2888         _prop
2889       } {
2890         origname = #2,
2891         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2892         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2893         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2894         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2895         content    = \prop_item:cn { \l_tmpa_str } { content } ,
2896         file       = \prop_item:cn { \l_tmpa_str } { file } ,
2897         lang       = \prop_item:cn { \l_tmpa_str } { lang } ,
2898         sig        = \prop_item:cn { \l_tmpa_str } { sig } ,
2899         meta       = \prop_item:cn { \l_tmpa_str } { meta } ,
2900         feature    = \prop_item:cn { \l_tmpa_str } { feature }
2901       }
2902     }

```

```

2903 } {
2904     \end{stex_annotate_env}
2905 }
2906 }
2907

```

25.2 Features

structure

```

2908
2909 \prop_new:N \l_stex_all_structures_prop
2910
2911 \keys_define:nn { stex / features / structure } {
2912     name .str_set_x:N = \l__stex_features_structure_name_str ,
2913 }
2914
2915 \cs_new_protected:Nn \__stex_features_structure_args:n {
2916     \str_clear:N \l__stex_features_structure_name_str
2917     \keys_set:nn { stex / features / structure } { #1 }
2918 }
2919
2920 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2921 % \__stex_features_structure_args:n { ##1 }
2922 % \str_if_empty:NT \l__stex_features_structure_name_str {
2923 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2924 % }
2925 %} {
2926 %
2927 %}
2928
2929 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2930     \__stex_features_structure_args:n { #1 }
2931     \str_if_empty:NT \l__stex_features_structure_name_str {
2932         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2933     }
2934     \exp_args:Nnnx
2935     \begin{structural@feature}{ structure }
2936         { \l__stex_features_structure_name_str }{}
2937         \seq_clear:N \l_tmpa_seq
2938         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2939     }
2940     {
2941         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2942         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2943         \str_set:Nx \l_tmpa_str {
2944             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2945             \prop_item:Nn \l_stex_current_module_prop { name }
2946         }
2947         \seq_map_inline:Nn \l_tmpa_seq {
2948             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2949         }
2950         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2951         \exp_args:Nnx

```



```

2952 \AddToHookNext { env / mathstructure / after }{
2953 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2954 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2955 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2956 \STEXexport {
2957 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2958 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2959 {\l_tmpa_str}
2960 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2961 {#2}{\l_tmpa_str}
2962 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2963 % \prop_item:Nn \l_stex_current_module_prop { origname },
2964 % \l_tmpa_str
2965 % }
2966 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2967 % #2,\l_tmpa_str
2968 % }
2969 % \tl_set:cx { #2 } {
2970 % \stex_invoke_structure:n { \l_tmpa_str }
2971 }
2972 }
2973
2974 \end{structural@feature}
2975 % \g_stex_last_feature_prop
2976 }

\instantiate
2977 \seq_new:N \l__stex_features_structure_field_seq
2978 \str_new:N \l__stex_features_structure_field_str
2979 \str_new:N \l__stex_features_structure_def_tl
2980 \prop_new:N \l__stex_features_structure_prop
2981 \NewDocumentCommand \instantiate { m O{} m }{
2982 \stex_smsmode_set_codes:
2983 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2984 \prop_set_eq:Nc \l__stex_features_structure_prop {
2985 c_stex_feature_\l_tmpa_str _prop
2986 }
2987 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2988 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2989 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2990 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2991 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2992 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2993 {!} \l_tmpa_tl
2994 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2995 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2996 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2997 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2998 }{
2999 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3000 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3001 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3002 \l_tmpa_tl
3003 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

3004         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3005         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3006     }{
3007         \tl_clear:N \l_tmpb_tl
3008     }
3009 }
3010 }{
3011     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3012     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3013         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3014         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3015         \tl_clear:N \l_tmpa_tl
3016     }{
3017         % TODO throw error
3018     }
3019 }
3020 % \l_tmpa_str: name
3021 % \l_tmpa_tl: definiens
3022 % \l_tmpb_tl: notation
3023 \tl_if_empty:NT \l__stex_features_structure_field_str {
3024     % TODO throw error
3025 }
3026 \str_clear:N \l_tmpb_str
3027
3028 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3029 \seq_map_inline:Nn \l_tmpa_seq {
3030     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3031     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3032     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3033         \seq_map_break:n {
3034             \str_set:Nn \l_tmpb_str { ####1 }
3035         }
3036     }
3037 }
3038 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3039     \l_tmpb_str
3040
3041 \tl_if_empty:NTF \l_tmpb_tl {
3042     \tl_if_empty:NF \l_tmpa_tl {
3043         \exp_args:Nx \use:n {
3044             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3045         }
3046     }
3047 }{
3048     \tl_if_empty:NTF \l_tmpa_tl {
3049         \exp_args:Nx \use:n {
3050             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3051         }
3052     }
3053 }{
3054     \exp_args:Nx \use:n {
3055         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3056         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3057     }

```

```

3058     }
3059   }
3060   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3061   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3062   % #3/\l_stex_features_structure_field_str
3063   % \par
3064   % \expandafter\present\csname
3065   %   g_stex_symdecl_
3066   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3067   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3068   %   #3/\l_stex_features_structure_field_str
3069   %   _prop
3070   % \endcsname
3071 }
3072
3073 \tl_clear:N \l__stex_features_structure_def_tl
3074
3075 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3076 \seq_map_inline:Nn \l_tmpa_seq {
3077   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3078   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3079   \exp_args:Nx \use:n {
3080     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3081
3082     }
3083   }
3084
3085   \prop_if_exist:cF {
3086     g_stex_symdecl_
3087     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3088     \prop_item:Nn \l_stex_current_module_prop {name} ?
3089     #3/\l_tmpa_str
3090     _prop
3091   }{
3092     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3093     \l_tmpb_str
3094     \exp_args:Nx \use:n {
3095       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3096     }
3097   }
3098 }
3099
3100 \symdecl*[type={\STEXsymbol{module-type}}{
3101   \_stex_term_math_oms:nnnn {
3102     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3103     \prop_item:Nn \l__stex_features_structure_prop {name}
3104     }{}{0}{}
3105   }{}{#3}
3106
3107   % TODO: -> sms file
3108
3109   \tl_set:cx{ #3 }{
3110     \stex_invoke_structure:nnn {
3111       \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3112     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3113   } {
3114     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3115     \prop_item:Nn \l__stex_features_structure_prop {name}
3116   }
3117 }
3118
3119 }

```

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

\stex_invoke_structure:nnn

```

3120 % #1: URI of the instance
3121 % #2: URI of the instantiated module
3122 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3123   \tl_if_empty:nTF{ #3 }{
3124     \prop_set_eq:Nc \l__stex_features_structure_prop {
3125       c_stex_feature_ #2 _prop
3126     }
3127     \tl_clear:N \l_tmpa_tl
3128     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3129     \seq_map_inline:Nn \l_tmpa_seq {
3130       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3131       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3132       \cs_if_exist:cT {
3133         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3134       }{
3135         \tl_if_empty:NF \l_tmpa_tl {
3136           \tl_put_right:Nn \l_tmpa_tl {,}
3137         }
3138         \tl_put_right:Nx \l_tmpa_tl {
3139           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3140         }
3141       }
3142     }
3143     \exp_args:No \mathstrut \l_tmpa_tl
3144   }{
3145     \stex_invoke_symbol:n{#1/#3}
3146   }
3147 }

```

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

```

3148 </package>

```

Chapter 26

STEX -Statements Implementation

```
3149 <*package>
3150
3151 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3152
3153 \protected\def\ignorespacesandpars{
3154   \begingroup\catcode13=10\relax
3155   \@ifnextchar\par{
3156     \endgroup\expandafter\ignorespacesandpars\@gobble
3157   }{
3158     \endgroup
3159   }
3160 }
3161
3162 <@@=stex_statements>
3163
3164   Warnings and error messages
3165
3166 \def\titleemph#1{\textbf{#1}}
3167
symboldoc
3168 \NewDocumentEnvironment{symboldoc}{m}{
3169   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3170   \seq_clear:N \l_tmpb_seq
3171   \seq_map_inline:Nn \l_tmpa_seq {
3172     \str_if_eq:nnF{ ##1 }{}{
3173       \stex_get_symbol:n { ##1 }
3174       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3175         \l_stex_get_symbol_uri_str
3176       }
3177     }
3178   }
3179   \par
3180   \exp_args:Nnnx
3181   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3182 }
```

```

3180 \end{stex_annotate_env}
3181 }

3182 \seq_new:N \g_stex_statements_patched_seq
3183
3184 \cs_new_protected:Nn \stex_statements_set_patched:n {
3185   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3186 }
3187
3188 \cs_new_protected:Nn \stex_statements_patch:nn {
3189   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3190     \AddToHook{begindocument}{
3191       \cs_if_exist:cTF{end#1}{
3192         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3193         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3194       }{
3195         \NewDocumentEnvironment{#1}{0{}}{
3196           \use:c{__stex_statements_#2_begin:n}{ }
3197         }{
3198           \use:c{__stex_statements_#2_end:}
3199         }
3200       }
3201     }
3202   }
3203 }

```

26.1 Definitions

definition

```

3204 \keys_define:nn {stex / definiendum }{
3205   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3206   root      .str_set:N     = \l__stex_statements_definiendum_root_str
3207 }
3208 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3209   \str_clear:N \l__stex_statements_definiendum_root_str
3210   \tl_clear:N \l__stex_statements_definiendum_post_tl
3211   \keys_set:nn { stex / definiendum }{ #1 }
3212 }
3213 \NewDocumentCommand \definiendum { O{} m m } {
3214   \__stex_statements_definiendum_args:n { #1 }
3215   \stex_get_symbol:n { #2 }
3216   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3217   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3218     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3219       \tl_set:Nn \l_tmpa_tl { #3 }
3220     } {
3221       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3222       \tl_set:Nn \l_tmpa_tl {
3223         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3224       }
3225     }
3226   } {
3227     \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3228 }
3229
3230 % TODO root
3231 \rustex_if:TF {
3232   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3233 } {
3234   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3235 }
3236 }
3237 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3238
3239 \NewDocumentCommand \definame { 0{ } m } {
3240   \__stex_statements_definiendum_args:n { #1 }
3241   % TODO: root
3242   \stex_get_symbol:n { #2 }
3243   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3244   \str_set:Nx \l_tmpa_str {
3245     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3246   }
3247   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3248   \rustex_if:TF {
3249     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3250       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3251     }
3252   } {
3253     \defemph@uri {
3254       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3255     } { \l_stex_get_symbol_uri_str }
3256   }
3257 }
3258 \stex_deactivate_macro:Nn \definame {definition~environments}
3259
3260 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3261   \stex_reactivate_macro:N \definiendum
3262   \stex_reactivate_macro:N \definame
3263   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3264   \seq_clear:N \l_tmpb_seq
3265   \seq_map_inline:Nn \l_tmpa_seq {
3266     \str_if_eq:nnF{ ##1 }{}{
3267       \stex_get_symbol:n { ##1 }
3268       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3269         \l_stex_get_symbol_uri_str
3270       }
3271     }
3272   }
3273   \stex_smsmode_set_codes:
3274   \exp_args:Nnnx
3275   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3276 }
3277
3278 \cs_new_protected:Nn \__stex_statements_defi_end: {
3279   \end{stex_annotate_env}
3280 }

```

Hook:

```
3281 \stex_statements_patch:nn{definition}{defi}
      inline:
3282 \NewDocumentCommand \inlinedef { m } {
3283   \begingroup
3284   \stex_reactivate_macro:N \definiendum
3285   \stex_reactivate_macro:N \definame
3286   \stex_ref_new_doc_target:n{}
3287   #1
3288   \endgroup
3289 }
```

26.2 Assertions

assertion

```
3290 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3291   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3292   \seq_clear:N \l_tmpb_seq
3293   \seq_map_inline:Nn \l_tmpa_seq {
3294     \str_if_eq:nnF{ ##1 }{}{
3295       \stex_get_symbol:n { ##1 }
3296       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3297         \l_stex_get_symbol_uri_str
3298       }
3299     }
3300   }
3301   \titleemph{Assertion}~
3302   \stex_smsmode_set_codes:
3303   \exp_args:Nnnx
3304   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3305 }
3306
3307 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3308   \end{stex_annotate_env}
3309 }
```

Hook:

```
3310 \stex_statements_patch:nn{assertion}{assertion}
      inline:
3311 \NewDocumentCommand \inlineass { m } {
3312   \begingroup
3313   \stex_ref_new_doc_target:n{}
3314   #1
3315   \endgroup
3316 }
```

theorem

```
3317 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3318   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3319   \seq_clear:N \l_tmpb_seq
```



```

3320 \seq_map_inline:Nn \l_tmpa_seq {
3321   \str_if_eq:nnF{ ##1 }{}{
3322     \stex_get_symbol:n { ##1 }
3323     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3324       \l_stex_get_symbol_uri_str
3325     }
3326   }
3327 }
3328 \titleemph{Theorem}~
3329 \stex_smsmode_set_codes:
3330 \exp_args:Nnnx
3331 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3332 }
3333
3334 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3335   \end{stex_annotate_env}
3336 }

```

Hook:

```

3337 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3338 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3339   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3340   \seq_clear:N \l_tmpb_seq
3341   \seq_map_inline:Nn \l_tmpa_seq {
3342     \str_if_eq:nnF{ ##1 }{}{
3343       \stex_get_symbol:n { ##1 }
3344       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3345         \l_stex_get_symbol_uri_str
3346       }
3347     }
3348   }
3349   \titleemph{Lemma}~
3350   \stex_smsmode_set_codes:
3351   \exp_args:Nnnx
3352   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3353 }
3354
3355 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3356   \end{stex_annotate_env}
3357 }

```

Hook:

```

3358 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3359 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3360   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3361   \seq_clear:N \l_tmpb_seq
3362   \seq_map_inline:Nn \l_tmpa_seq {
3363     \str_if_eq:nnF{ ##1 }{}{
3364       \stex_get_symbol:n { ##1 }

```

```

3365     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3366       \l_stex_get_symbol_uri_str
3367     }
3368   }
3369 }
3370 \titleemph{Axiom}~
3371 \stex_smsmode_set_codes:
3372 \exp_args:Nnnx
3373 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3374 }
3375
3376 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3377   \end{stex_annotate_env}
3378 }

Hook:

3379 \stex_statements_patch:nn{axiom}{axiom}

```

26.3 Examples

example

```

3380 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3381   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3382   \seq_clear:N \l_tmpb_seq
3383   \seq_map_inline:Nn \l_tmpa_seq {
3384     \str_if_eq:nnF{ ##1 }{}{
3385       \stex_get_symbol:n { ##1 }
3386       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3387         \l_stex_get_symbol_uri_str
3388       }
3389     }
3390   }
3391   \titleemph{Example}~
3392   \stex_smsmode_set_codes:
3393   \exp_args:Nnnx
3394   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3395 }
3396
3397 \cs_new_protected:Nn \__stex_statements_example_end: {
3398   \end{stex_annotate_env}
3399 }

Hook:

3400 \stex_statements_patch:nn{example}{example}

inline:

3401 \NewDocumentCommand \inlineex { m } {
3402   \begingroup
3403   \stex_ref_new_doc_target:n{
3404     #1
3405   }
3406   \endgroup
}

```

26.4 OMText

```

3407 \keys_define:nn { stex / omtext} {
3408   id      .str_set_x:N    = \l_stex_omtext_id_str ,
3409   title   .tl_set:N       = \l_stex_omtext_title_tl ,
3410   type    .tl_set_x:N     = \l_stex_omtext_type_tl ,
3411   for     .tl_set_x:N     = \l_stex_omtext_for_tl ,
3412   from    .tl_set_x:N     = \l_stex_omtext_from_tl ,
3413   start   .tl_set:N       = \l_stex_omtext_start_tl ,
3414 }
3415 \cs_new_protected:Nn \stex_omtext_args:n {
3416   \tl_clear:N \l_stex_omtext_title_tl
3417   \tl_clear:N \l_stex_omtext_start_tl
3418   \keys_set:nn { stex / omtext }{ #1 }
3419 }
3420 \newif\if@in@omtext\@in@omtextfalse
3421 \NewDocumentEnvironment {omtext} { 0{} } {
3422   \stex_omtext_args:n { #1 }
3423   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3424     \tl_if_empty:NF \l_stex_omtext_title_tl {
3425       \titleemph{\l_stex_omtext_title_tl}:~
3426     }
3427   }{
3428     \titleemph{\l_stex_omtext_start_tl}~
3429   }
3430   \@in@omtexttrue
3431
3432   \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3433   \stex_smsmode_set_codes:
3434   \ignorespacesandpars
3435 }{}
3436 \</package>

```

Chapter 27

The Implementation

27.1 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).¹⁰

```
3437 <*package>
3438 <@@=stex_sproof>
3439
3440 %%%%%%%%%% sproof.dtx %%%%%%%%%%
3441
```

27.2 Proofs

We first define some keys for the proof environment.

```
3442 \keys_define:nn { stex / spf } {
3443   id          .str_set:N = \l__stex_sproof_spf_id_str,
3444   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3445   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3446   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3447   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3448   type       .tl_set:N  = \l__stex_sproof_spf_type_tl,
3449   title      .tl_set:N  = \l__stex_sproof_spf_title_tl,
3450   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3451   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3452   method     .tl_set:N  = \l__stex_sproof_spf_method_tl
3453 }
3454 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3455   \str_clear:N \l__stex_sproof_spf_id_str
3456   \tl_clear:N \l__stex_sproof_spf_display_tl
3457   \tl_clear:N \l__stex_sproof_spf_for_tl
3458   \tl_clear:N \l__stex_sproof_spf_from_tl
3459   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3460   \tl_clear:N \l__stex_sproof_spf_type_tl
3461   \tl_clear:N \l__stex_sproof_spf_title_tl

```

¹⁰EDNOTE: need an implementation for L^AT_EXML

```

3462 \tl_clear:N \l__stex_sproof_spf_continues_tl
3463 \tl_clear:N \l__stex_sproof_spf_functions_tl
3464 \tl_clear:N \l__stex_sproof_spf_method_tl
3465 \keys_set:nn { stex / spf }{ #1 }
3466 }

```

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

```

3467 \def\spf@flow{flow}

```

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

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 manages⁶ 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.

```

3468 \newcount\count_ten
3469 \newenvironment{pst@with@label}[1]{
3470   \edef\pst@label{#1}
3471   \advance\count_ten by 1\relax
3472   \count_ten=1
3473 }{
3474   \advance\count_ten by -1\relax
3475 }

```

`\the@pst@label` `\the@pst@label` evaluates to the current step label.

```

3476 \def\the@pst@label{
3477   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3478 }

```

(End definition for `\the@pst@label`. This function is documented on page ??.)

`\setpstlabelstyle` `\setpstlabelstyle{metaKey-Val pairs}` makes the labeling style customizable. `\setpstlabelstyle{pr}` will change the labeling style from **P.1.2.3** to **Pr-1-2-3†**. `\setpstlabelstyledefault` will set the labeling style back to default.

```

3479 \keys_define:nn { stex / pstlabel }{
3480   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3481   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3482   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3483 }
3484 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

⁶This gets the labeling right but only works 8 levels deep

```

3485 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3486 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3487 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3488 }
3489 \__stex_sproof_pstlabel_args:n {}
3490 \newcommand\setpstlabelstyle[1]{
3491   \__stex_sproof_pstlabel_args:n {#1}
3492 }
3493 \newcommand\setpstlabelstyledefault{%
3494   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3495 }

```

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

\pstlabelstyle \pstlabelstyle just sets the \pst@make@label macro according to the style.

```

3496 \ExplSyntaxOff
3497 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3498 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3499 \def\pst@make@label@short#1#2{#2}
3500 \def\pst@make@label@empty#1#2{}
3501 \ExplSyntaxOn
3502 \def\pstlabelstyle#1{%
3503   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3504 }%
3505 \pstlabelstyle{long}%

```

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

\next@pst@label \next@pst@label increments the step label at the current level.

```

3506 \def\next@pst@label{%
3507   \global\advance\count\count10 by 1%
3508 }%

```

(End definition for \next@pst@label. This function is documented on page ??.)

\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

```

3509 \def\sproof@box{
3510   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3511 }
3512 \def\spf@proofend{\sproof@box}
3513 \def\sproofend{
3514   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3515     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3516   }
3517 }
3518 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3519 \def\spf@proofsketch@kw{Proof Sketch}
3520 \def\spf@proof@kw{Proof}
3521 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3522 \cs_if_exist:NT \bbl@loaded {
3523   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3524   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3525     \input{proof-ngerman.lda}
3526   }
3527   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3528     \input{proof-finnish.lda}
3529   }
3530   \clist_if_in:NnT \l_tmpa_clist {french}{
3531     \input{proof-french.lda}
3532   }
3533   \clist_if_in:NnT \l_tmpa_clist {russian}{
3534     \input{proof-russian.lda}
3535   }
3536 }
3537

```

spfsketch

```

3538 \newcommand\spfsketch[2] [] {
3539   \__stex_sproof_spf_args:n{#1}
3540   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3541     \titleemph{
3542       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3543         \spf@proofsketch@kw
3544       }{
3545         \l__stex_sproof_spf_type_tl
3546       }
3547     }:
3548   }
3549   {-#2}
3550   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3551   \sproofend
3552 }

```

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

spfeq This is very similar to `\spfsketch`, but uses a computation array¹¹¹²

```

3553 \newenvironment{spfeq}[2] [] {
3554   \__stex_sproof_spf_args:n{#1}
3555   %\sref@target
3556   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3557     \titleemph{
3558       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3559         \spf@proof@kw
3560       }{
3561         \l__stex_sproof_spf_type_tl
3562       }
3563     }:

```

¹¹EDNOTE: This should really be more like a tabular with an ensuremath in it. or invoke text on the last column

¹²EDNOTE: document above

```

3564 }
3565 {~#2}
3566 \begin{displaymath}\begin{array}{rcll}
3567 }{
3568 \end{array}\end{displaymath}
3569 }

```

(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.

```

3570 \newenvironment{spf@proof}[2][]{
3571   \__stex_sproof_spf_args:n{#1}
3572   %\sref@target
3573   \count_ten=10
3574   \par\noindent
3575   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3576     \titleemph{
3577       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3578         \spf@proof@kw
3579       }{
3580         \l__stex_sproof_spf_type_tl
3581       }
3582     }:
3583   }
3584   {~#2}
3585   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3586   \def\pst@label{}
3587   \newcount\pst@count% initialize the labeling mechanism
3588   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3589   }{
3590     \end{pst@with@label}\end{description}
3591   }
3592   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3593   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3594 \newcommand\spfidea[2][]{
3595   \__stex_sproof_spf_args:n{#1}
3596   \titleemph{
3597     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3598       \l__stex_sproof_spf_type_tl
3599     }:
3600   }~#2
3601   \sproofend
3602 }

```

(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 13

```

3603 \newenvironment{spfstep}[1][]{
3604   \_stex_sproof_spf_args:n{#1}
3605   \@in@omtexttrue
3606   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3607     \item[\the@pst@label]
3608   }
3609   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3610     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3611   }
3612   %\sref@label@id{\pst@label}
3613   \ignorespacesandpars
3614 }{
3615   \next@pst@label\ignorespacesandpars
3616 }

```

sproofcomment

```

3617 \newenvironment{sproofcomment}[1][]{
3618   \_stex_sproof_spf_args:n{#1}
3619   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3620     \item[\the@pst@label]
3621   }
3622 }{
3623   \next@pst@label
3624 }

```

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) `proproofof` environment is started.

```

3625 \newenvironment{subproof}[2][]{
3626   \_stex_sproof_spf_args:n{#1}
3627   \def\@test{#2}
3628   \ifx\@test\empty\else
3629     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3630       \item[\the@pst@label]
3631     }{#2}
3632   \fi
3633   \begin{pst@with@label}{\pst@label,\number\count_ten}
3634 }{
3635   \end{pst@with@label}\next@pst@label
3636 }

```

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

```

3637 \newenvironment{spfcases}[2][]{
3638   \def\@test{#1}
3639   \ifx\@test\empty
3640     \begin{subproof}[method=by-cases]{#2}
3641   \else
3642     \begin{subproof}[#1,method=by-cases]{#2}
3643   \fi
3644 }{

```

¹³EdNOTE: MK: labeling of steps does not work yet.

```

3645 \end{subproof}
3646 }

```

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

```

3647 \newenvironment{spfcase}[2] [] {
3648   \__stex_sproof_spf_args:n{#1}
3649   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3650     \item[\the@pst@label]
3651   }
3652   \def\@test{#2}
3653   \ifx\@test\@empty
3654     \else
3655       {\titleemph{#2}:~}
3656     \fi
3657   \begin{pst@with@label}{\pst@label,\number\count_ten}
3658 }{
3659   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3660     \sproofend
3661   }
3662   \end{pst@with@label}
3663   \next@pst@label
3664 }

```

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

```

3665 \newcommand\spfcasesketch[3] [] {
3666   \__stex_sproof_spf_args:n{#1}
3667   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3668     \item[\the@pst@label]
3669   }
3670   \def\@test{#2}
3671   \ifx\@test\@empty
3672     \else
3673       {\titleemph{#2}:~}
3674     \fi#3
3675   \next@pst@label
3676 }%

```

27.3 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.

```

3677 \keys_define:nn { stex / just }{
3678   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3679   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
3680   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
3681   args        .tl_set:N   = \l__stex_sproof_just_args_tl
3682 }

```

The next three environments and macros are purely semantic, so we ignore the keyval arguments for now and only display the content.¹⁴

¹⁴EDNOTE: need to do something about the premise in draft mode.

justification

```
3683 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3684 \newcommand\premise[2] [] {#2}
```

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

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

```
3685 \newcommand\justarg[2] [] {#2}
```

```
3686 \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 28

STEX -Others Implementation

```
3687 <*package>
3688
3689 %%%%%%%%%% others.dtx %%%%%%%%%%
3690
3691 <@@=stex_others>
    Warnings and error messages
3692 % None

\MSC Math subject classifier

3693 \NewDocumentCommand \MSC {m} {
3694 % TODO
3695 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3696 \@ifpackageloaded{tikzinput}{
3697 \RequirePackage{stex-tikzinput}
3698 }{}
3699 </package>
```

Chapter 29

STEX -Metatheory Implementation

```
3700 <*package>
3701 <@@=stex_modules>
3702
3703 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3704
3705 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3706 \begingroup
3707 \stex_module_setup:nn{
3708   ns=\c_stex_metatheory_ns_str,
3709   meta=NONE
3710 }{Metatheory}
3711 \stex_reactivate_macro:N \symdecl
3712 \stex_reactivate_macro:N \notation
3713 \stex_reactivate_macro:N \symdef
3714 \ExplSyntaxOff
3715 \csname stex_suppress_html:n\endcsname{
3716   % is-a (a:A, a \in A, a is an A, etc.)
3717   \symdecl[args=ai]{isa}
3718   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3719   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3720   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3721
3722   % bind (\forall, \Pi, \lambda etc.)
3723   \symdecl[args=Bi]{bind}
3724   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3725   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3726   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; )}{#1 \comp, #2}
3727
3728   % dummy variable
3729   \symdecl{dummyvar}
3730   \notation[underscore]{dummyvar}{\comp\_}
3731   \notation[dot]{dummyvar}{\comp\cdot}
3732   \notation[dash]{dummyvar}{\comp{\rm --}}
3733
3734   %fromto (function space, Hom-set, implication etc.)
```

```

3735 \symdecl[args=ai]{fromto}
3736 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3737 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3738
3739 % mapto (lambda etc.)
3740 %\symdecl[args=Bi]{mapto}
3741 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3742 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3743 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3744
3745 % function/operator application
3746 \symdecl[args=ia]{apply}
3747 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3748 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3749
3750 % ‘‘type’’ of all collections (sets,classes,types,kinds)
3751 \symdecl{collection}
3752 \notation[U]{collection}{\comp{\mathcal{U}}}
3753 \notation[set]{collection}{\comp{\textsf{Set}}}
3754
3755 % sequences
3756 \symdecl[args=1]{seqtype}
3757 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3758
3759 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3760 \notation[ui]{sequence-index}{#1^{#2}}
3761
3762 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
3763 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
3764 % ^ superceded by \aseqfromto and \livar/\uivar
3765
3766 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3767 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses},#2}{#1\comp,#2}
3768 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
3769
3770 % letin (‘‘let’’, local definitions, variable substitution)
3771 \symdecl[args=bii]{letin}
3772 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3773 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3774 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3775
3776 % structures
3777 \symdecl*[args=1]{module-type}
3778 \notation{module-type}{\mathtt{MOD} #1}
3779 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3780 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3781
3782 }
3783 \ExplSyntaxOn
3784 \stex_add_to_current_module:n{
3785   \let\nappa\apply
3786   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3787   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
3788   \def\livar{\csname sequence-index\endcsname[li]}

```

```

3789 \def\uivar{\csname sequence-index\endcsname[ui]}
3790 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3791 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3792 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3793 }
3794 \__stex_modules_end_module:
3795 \endgroup
3796 \endpackage

```

Chapter 30

Tikzinput Implementation

```
3797 <*package>
3798
3799 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3800
3801 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3802 \RequirePackage{l3keys2e}
3803
3804 \keys_define:nn { tikzinput } {
3805   image .bool_set:N = \c_tikzinput_image_bool,
3806   image .default:n = false ,
3807   unknown .code:n = {}
3808 }
3809
3810 \ProcessKeysOptions { tikzinput }
3811
3812 \bool_if:NTF \c_tikzinput_image_bool {
3813   \RequirePackage{graphicx}
3814
3815   \providecommand\usetikzlibrary[]{}
3816   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3817 }{
3818   \RequirePackage{tikz}
3819   \RequirePackage{standalone}
3820
3821   \newcommand \tikzinput [2] [] {
3822     \setkeys{Gin}{#1}
3823     \ifx \Gin@ewidth \Gin@exclamation
3824       \ifx \Gin@eheight \Gin@exclamation
3825         \input { #2 }
3826       \else
3827         \resizebox{!}{ \Gin@eheight }{
3828           \input { #2 }
3829         }
3830       \fi
3831     \else
3832       \ifx \Gin@eheight \Gin@exclamation
3833         \resizebox{ \Gin@ewidth }{!}{
3834           \input { #2 }
```



```

3835     }
3836     \else
3837         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3838             \input { #2 }
3839         }
3840     \fi
3841 \fi
3842 }
3843 }
3844
3845 \newcommand \ctikzinput [2] [] {
3846     \begin{center}
3847         \tikzinput [1] {#2}
3848     \end{center}
3849 }
3850
3851 \@ifpackageloaded{stex}{
3852     \RequirePackage{stex-tikzinput}
3853 }{}
3854
3855 </package>
3856 <*stex>
3857 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3858 \RequirePackage{stex}
3859 \RequirePackage{tikzinput}
3860
3861 \newcommand\mhtikzinput [2] [] {%
3862     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3863     \stex_in_repository:nn\Gin@mhrepos{
3864         \tikzinput [1]{\mhpath{##1}{#2}}
3865     }
3866 }
3867 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
3868 </stex>

```

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

Chapter 31

document-structure.sty Implementation

31.1 The OMDoc Class

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

```
3869 \*cls)
3870 \@@=document_structure)
3871 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3872 \RequirePackage{l3keys2e,expl-keystr-compat}
```

31.2 Class Options

To initialize the `omdoc` 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`

```
3873 \keys_define:nn{ document-structure / pkg }{
3874   class      .str_set_x:N = \c_document_structure_class_str,
3875   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3876   report     .code:n      = {
3877     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3878     \str_set:Nn \c_document_structure_class_str {report}
3879   },
3880   book       .code:n      = {
3881     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3882     \str_set:Nn \c_document_structure_class_str {book}
3883   },
3884   bookpart   .code:n      = {
3885     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3886     \str_set:Nn \c_document_structure_class_str {book}
3887     \str_set:Nn \c_document_structure_topsect_str {chapter}
3888   },
```

```

3889 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3890 unknown     .code:n      = {
3891   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3892 }
3893 }
3894 \ProcessKeysOptions{ document-structure / pkg }
3895 \str_if_empty:NT \c_document_structure_class_str {
3896   \str_set:Nn \c_document_structure_class_str {article}
3897 }
3898 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3899   {\c_document_structure_class_str}
3900

```

31.3 Beefing up the document environment

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

```

3901 \RequirePackage{omdoc}
3902 \bool_if:NF \c_document_structure_minimal_bool {
3903   \RequirePackage{stex-compatibility}

```

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

document For the moment we do not use them on the L^AT_EX level, but the document identifier is picked up by L^AT_EXML.¹⁵

```

3904 \keys_define:nn { document-structure / document }{
3905   id .str_set_x:N = \c_document_structure_document_id_str
3906 }
3907 \let\__document_structure_orig_document=\document
3908 \renewcommand{\document}[1][]{
3909   \keys_set:nn{ document-structure / document }{ #1 }
3910   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3911   \__document_structure_orig_document
3912 }

```

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

```

3913 }
3914 \</cls>

```

31.4 Implementation: OMDoc Package

```

3915 \*package>
3916 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3917 \RequirePackage{expl-keystr-compat,13keys2e}

```

31.5 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).

¹⁵EdNOTE: faking documentkeys for now. @HANG, please implement

```

3918
3919 \keys_define:nn{ document-structure / pkg }{
3920   class      .str_set_x:N = \c_document_structure_class_str,
3921   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3922   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3923 }
3924 \ProcessKeysOptions{ document-structure / pkg }
3925 \str_if_empty:NT \c_document_structure_class_str {
3926   \str_set:Nn \c_document_structure_class_str {article}
3927 }
3928 \str_if_empty:NT \c_document_structure_topsect_str {
3929   \str_set:Nn \c_document_structure_topsect_str {section}
3930 }

```

Then we need to set up the packages by requiring the `sref` package to be loaded.

```

3931 \RequirePackage{xspace}
3932 \RequirePackage{comment}
3933 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

We set up triggers for the other languages, currently only German.

```

3934 \@ifpackageloaded{babel}{
3935   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3936   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3937     \input{omdoc-ngerman.ldf}
3938   }
3939 }{}
3940 %\AfterBabelLanguage{ngerman}{\input{omdoc-ngerman.ldf}}

```

`\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.

```

3941 \int_new:N \l_document_structure_section_level_int
3942 \str_case:VnF \c_document_structure_topsect_str {
3943   {part}{
3944     \int_set:Nn \l_document_structure_section_level_int {0}
3945   }
3946   {chapter}{
3947     \int_set:Nn \l_document_structure_section_level_int {1}
3948   }
3949 }{
3950   \str_case:VnF \c_document_structure_class_str {
3951     {book}{
3952       \int_set:Nn \l_document_structure_section_level_int {0}
3953     }
3954     {report}{
3955       \int_set:Nn \l_document_structure_section_level_int {0}
3956     }
3957   }{
3958     \int_set:Nn \l_document_structure_section_level_int {2}
3959   }
3960 }

```

31.6 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.¹⁶

EdN:16

```
3961 \def\current@section@level{document}%
3962 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3963 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3964 \cs_new_protected:Npn \skipomgroup {
3965   \ifcase\l_document_structure_section_level_int
3966   \or\stepcounter{part}
3967   \or\stepcounter{chapter}
3968   \or\stepcounter{section}
3969   \or\stepcounter{subsection}
3970   \or\stepcounter{subsubsection}
3971   \or\stepcounter{paragraph}
3972   \or\stepcounter{subparagraph}
3973   \fi
3974 }
```

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

`blindomgroup`

```
3975 \newcommand\at@begin@blindomgroup[1]{%
3976 \newenvironment{blindomgroup}
3977 {
3978   \int_incr:N\l_document_structure_section_level_int
3979   \at@begin@blindomgroup\l_document_structure_section_level_int
3980 }{}}
```

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

```
3981 \newcommand\omgroup@nonum[2]{
3982   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3983   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3984 }
```

(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.

```
3985 \newcommand\omgroup@num[2]{
```

¹⁶EDNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

```

3986 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3987   \@nameuse{#1}{#2}
3988 }{
3989   \cs_if_exist:NTF\rdfmata@sectioning{
3990     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3991   }{
3992     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3993   }
3994 }
3995 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3996 }

```

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

omgroup

```

3997 \keys_define:nn { document-structure / omgroup }{
3998   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3999   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4000   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4001   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4002   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4003   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4004   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4005   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4006   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4007   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4008 }
4009 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4010   \str_clear:N \l__document_structure_omgroup_id_str
4011   \str_clear:N \l__document_structure_omgroup_date_str
4012   \clist_clear:N \l__document_structure_omgroup_creators_clist
4013   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4014   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4015   \tl_clear:N \l__document_structure_omgroup_type_tl
4016   \tl_clear:N \l__document_structure_omgroup_short_tl
4017   \tl_clear:N \l__document_structure_omgroup_display_tl
4018   \tl_clear:N \l__document_structure_omgroup_intro_tl
4019   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4020   \keys_set:nn { document-structure / omgroup } { #1 }
4021 }

```

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.

```

4022 \newif\if@mainmatter\@mainmattertrue
4023 \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.

```

4024 \keys_define:nn { document-structure / sectioning }{
4025   name .str_set_x:N = \l__document_structure_sect_name_str ,
4026   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4027   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4028   num .bool_set:N = \l__document_structure_sect_num_bool ,
4029 }

```

```

4030 \cs_new_protected:Nn \__document_structure_sect_args:n {
4031   \str_clear:N \l__document_structure_sect_name_str
4032   \str_clear:N \l__document_structure_sect_ref_str
4033   \bool_set_false:N \l__document_structure_sect_clear_bool
4034   \bool_set_false:N \l__document_structure_sect_num_bool
4035   \keys_set:nn { document-structure / sectioning } { #1 }
4036 }
4037 \newcommand\omdoc@sectioning[3][]{
4038   \__document_structure_sect_args:n {#1}
4039   \let\omdoc@sect@name\l__document_structure_sect_name_str
4040   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4041   \if@mainmatter% numbering not overridden by frontmatter, etc.
4042     \bool_if:NTF \l__document_structure_sect_num_bool {
4043       \omgroup@num{#2}{#3}
4044     }{
4045       \omgroup@nonum{#2}{#3}
4046     }
4047     \def\current@section@level{\omdoc@sect@name}
4048   \else
4049     \omgroup@nonum{#2}{#3}
4050   \fi
4051 }% 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.

```

4052 \newcommand\omgroup@redefine@addtocontents[1]{%
4053   %\edef\__document_structureimport{#1}%
4054   %\@for\@I:=\__document_structureimport\do{%
4055     %\edef\@path{\csname module@\@I @path\endcsname}%
4056     %\@ifundefined{tf@toc}\relax%
4057     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4058   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4059   %\def\addcontentsline##1##2##3{%
4060     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4061   %\else% hyperref.sty not loaded
4062   %\def\addcontentsline##1##2##3{%
4063     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4064   %\fi
4065 }% hyperref.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.

```

4066 \int_new:N \l_document_structure_omgroup_level_int
4067 \newenvironment{omgroup}[2][]{% keys, title
4068 {
4069   \__document_structure_omgroup_args:n { #1 }%\sref@target%

```

If the `loadmodules` key is set on `\begin{omgroup}`, we redefine the `\addcontetsline` macro that determines how the sectioning commands below construct the entries for the table of contents.

```

4070 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4071   \omgroup@redefine@addtocontents{
4072     %\@ifundefined{module@id}\used@modules%
4073     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4074     }
4075 }

```

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

```

4076 \int_incr:N \l_document_structure_omgroup_level_int
4077 \int_incr:N \l_document_structure_section_level_int
4078 \ifcase\l_document_structure_section_level_int
4079   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4080   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4081   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4082   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4083   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4084   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4085   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4086 \fi
4087 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4088 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4089 }% for customization
4090 {}

```

and finally, we localize the sections

```

4091 \newcommand\omdoc@part@kw{Part}
4092 \newcommand\omdoc@chapter@kw{Chapter}
4093 \newcommand\omdoc@section@kw{Section}
4094 \newcommand\omdoc@subsection@kw{Subsection}
4095 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4096 \newcommand\omdoc@paragraph@kw{paragraph}
4097 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

31.7 Front and Backmatter

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

`\printindex`

```

4098 \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).

```

4099 \cs_if_exist:NTF\frontmatter{
4100   \let\__document_structure_orig_frontmatter\frontmatter
4101   \let\frontmatter\relax
4102 }{
4103   \tl_set:Nn\__document_structure_orig_frontmatter{
4104     \clearpage
4105     \@mainmatterfalse
4106     \pagenumbering{roman}
4107   }
4108 }
4109 \cs_if_exist:NTF\backmatter{

```



```

4110 \let\__document_structure_orig_backmatter\backmatter
4111 \let\backmatter\relax
4112 }{
4113 \tl_set:Nn\__document_structure_orig_backmatter{
4114 \clearpage
4115 \@mainmatterfalse
4116 \pagenumbering{roman}
4117 }
4118 }

```

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.

```

4119 \newenvironment{frontmatter}{
4120 \__document_structure_orig_frontmatter
4121 }{
4122 \cs_if_exist:NTF\mainmatter{
4123 \mainmatter
4124 }{
4125 \clearpage
4126 \@mainmattertrue
4127 \pagenumbering{arabic}
4128 }
4129 }

```

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

```

4130 \newenvironment{backmatter}{
4131 \__document_structure_orig_backmatter
4132 }{
4133 \cs_if_exist:NTF\mainmatter{
4134 \mainmatter
4135 }{
4136 \clearpage
4137 \@mainmattertrue
4138 \pagenumbering{arabic}
4139 }
4140 }

```

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

```

4141 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4142 \newcommand\afterprematurestop{}
4143 \def\prematurestop@endomgroup{
4144 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4145 \end{omgroup}
4146 \int_decr:N \l_document_structure_omgroup_level_int
4147 \prematurestop@endomgroup
4148 }
4149 }
4150 \providecommand\prematurestop{

```

```

4151 \message{Stopping sTeX processing prematurely}
4152 \prematurestop@endomgroup
4153 \afterprematurestop
4154 \end{document}
4155 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

4156 \RequirePackage{etoolbox}
4157 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4158 \newrobustcmd\useSGvar[1]{%
4159 \@ifundefined{sTeX@Gvar@#1}
4160 {\PackageError{omdoc}
4161 {The sTeX Global variable #1 is undefined}
4162 {set it with \protect\setSGvar}}
4163 \@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.

```

4164 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4165 \@ifundefined{sTeX@Gvar@#1}
4166 {\PackageError{omdoc}
4167 {The sTeX Global variable #1 is undefined}
4168 {set it with \protect\setSGvar}}
4169 {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 32

MiKoSlides – Implementation

32.1 Class and Package Options

We define some Package Options and switches for the `mikoslides` class and activate them by passing them on to `beamer.cls` and `omdoc.cls` and the `mikoslides` 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.

```
4170 \*cls)
4171 \@@=mikoslides)
4172 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4173 \RequirePackage{l3keys2e,expl-keystr-compat}
4174
4175 \keys_define:nn{mikoslides / cls}{
4176   class .code:n = {
4177     \PassOptionsToClass{\CurrentOption}{omdoc}
4178     \str_if_eq:nnT{#1}{book}{
4179       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4180     }
4181     \str_if_eq:nnT{#1}{report}{
4182       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4183     }
4184   },
4185   notes .bool_set:N = \c__mikoslides_notes_bool ,
4186   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4187   unknown .code:n = {
4188     \PassOptionsToClass{\CurrentOption}{omdoc}
4189     \PassOptionsToClass{\CurrentOption}{beamer}
4190     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4191   }
4192 }
4193 \ProcessKeysOptions{ mikoslides / cls }
4194 \bool_if:NTF \c__mikoslides_notes_bool {
4195   \PassOptionsToPackage{notes=true}{mikoslides}
4196 }{
4197   \PassOptionsToPackage{notes=false}{mikoslides}
4198 }
4199 \</cls)
```

now we do the same for the mikoslides package.

```

4200 <*package>
4201 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4202 \RequirePackage{l3keys2e,expl-keystr-compat}
4203
4204 \keys_define:nn{mikoslides / pkg}{
4205   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4206   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4207   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4208   slides        .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4209   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4210   frameimages .bool_set:N = \c__mikoslides_frameimages_bool ,
4211   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4212   nopproblems .bool_set:N = \c__mikoslides_nopproblems_bool,
4213   unknown      .code:n      = {
4214     \PassOptionsToClass{\CurrentOption}{stex}
4215     \PassOptionsToClass{\CurrentOption}{tikzinput}
4216   }
4217 }
4218 \ProcessKeysOptions{ mikoslides / pkg }
4219 \newif\ifnotes
4220 \bool_if:NTF \c__mikoslides_notes_bool {
4221   \notesttrue
4222 }{
4223   \notesfalse
4224 }
4225

```

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

```

4226 \str_if_empty:NTF \c__mikoslides_topsect_str {
4227   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4228 }{
4229   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4230 }
4231 </package>

```

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

```

4232 <*cls>
4233 \bool_if:NTF \c__mikoslides_notes_bool {
4234   \LoadClass{omdoc}
4235 }{
4236   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4237   \newcounter{Item}
4238   \newcounter{paragraph}
4239   \newcounter{subparagraph}
4240   \newcounter{Hfootnote}
4241   \RequirePackage{omdoc}
4242 }

```

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

```

4243 \RequirePackage{mikoslides}
4244 </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 `mikoslides.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.

```

4245 \*package>
4246 \bool_if:NT \c__mikoslides_notes_bool {
4247   \RequirePackage{a4wide}
4248   \RequirePackage{marginnote}
4249   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4250   \RequirePackage{mdframed}
4251   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4252   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4253 }
4254 \RequirePackage{stex-compatibility}
4255 \RequirePackage{stex-tikzinput}
4256 \RequirePackage{etoolbox}
4257 \RequirePackage{amssymb}
4258 \RequirePackage{amsmath}
4259 \RequirePackage{comment}
4260 \RequirePackage{textcomp}
4261 \RequirePackage{url}
4262 \RequirePackage{graphicx}
4263 \RequirePackage{pgf}

```

32.2 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`.¹⁷

```

4264 \bool_if:NT \c__mikoslides_notes_bool {
4265   \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
4266 }

```

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

```

4267 \newcounter{slide}
4268 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4269 \newlength{\slideheight}\setlength{\slideheight}{9cm}

```

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.

```

4270 \bool_if:NTF \c__mikoslides_notes_bool {
4271   \renewenvironment{note}{\ignorespaces}{\}
4272 }{
4273   \excludecomment{note}
4274 }

```

¹⁷EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

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.

```
4275 \bool_if:NT \c__mikoslides_notes_bool {
4276   \newlength{\slideframewidth}
4277   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4278 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4279   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4280     \bool_set_true:N #1
4281   }{
4282     \bool_set_false:N #1
4283   }
4284 }
4285 \keys_define:nn{mikoslides / frame}{
4286   label .str_set_x:N = \l__mikoslides_frame_label_str,
4287   allowframebreaks .code:n = {
4288     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4289   },
4290   allowdisplaybreaks .code:n = {
4291     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4292   },
4293   fragile .code:n = {
4294     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4295   },
4296   shrink .code:n = {
4297     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4298   },
4299   squeeze .code:n = {
4300     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4301   },
4302   t .code:n = {
4303     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4304   },
4305 }
4306 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4307   \str_clear:N \l__mikoslides_frame_label_str
4308   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4309   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4310   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4311   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4312   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4313   \bool_set_true:N \l__mikoslides_frame_t_bool
4314   \keys_set:nn { mikoslides / frame }{ #1 }
4315 }
```

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

```
4316 \renewenvironment{frame}[1][]{
4317   \__mikoslides_frame_args:n{#1}
4318   \sffamily
4319   \stepcounter{slide}
4320   \def\@currentlabel{\theslide}
4321   \str_if_empty:NF \l__mikoslides_frame_label_str {
4322     \label{\l__mikoslides_frame_label_str}
```

4323 }

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

```

4324 \def\itemize@level{outer}
4325 \def\itemize@outer{outer}
4326 \def\itemize@inner{inner}
4327 \renewcommand\newpage{\addtocounter{framenum}{1}}
4328 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4329 \renewenvironment{itemize}{
4330   \ifx\itemize@level\itemize@outer
4331     \def\itemize@label{$\rhd$}
4332   \fi
4333   \ifx\itemize@level\itemize@inner
4334     \def\itemize@label{$\scriptstyle\rhd$}
4335   \fi
4336   \begin{list}
4337   {\itemize@label}
4338   {\setlength{\labelsep}{.3em}
4339    \setlength{\labelwidth}{.5em}
4340    \setlength{\leftmargin}{1.5em}
4341   }
4342   \edef\itemize@level{\itemize@inner}
4343 }{
4344   \end{list}
4345 }

```

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

```

4346 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4347 }{
4348   \medskip\miko@slidelabel\end{mdframed}
4349 }

```

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

`\frametitle`

```

4350 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4351 }

```

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

EdN:18

`\pause` 18

```

4352 \bool_if:NT \c__mikoslides_notes_bool {
4353   \newcommand\pause{}
4354 }

```

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

`nomtext`

```

4355 \bool_if:NTF \c__mikoslides_notes_bool {
4356   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}
4357 }{
4358   \excludecomment{nomtext}
4359 }

```

¹⁸EdNOTE: MK: fake it in notes mode for now

nomgroup

```
4360 \bool_if:NTF \c__mikoslides_notes_bool {
4361   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4362 }{
4363   \excludecomment{nomgroup}
4364 }
```

ndefinition

```
4365 \bool_if:NTF \c__mikoslides_notes_bool {
4366   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4367 }{
4368   \excludecomment{ndefinition}
4369 }
```

nassertion

```
4370 \bool_if:NTF \c__mikoslides_notes_bool {
4371   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4372 }{
4373   \excludecomment{nassertion}
4374 }
```

nsproof

```
4375 \bool_if:NTF \c__mikoslides_notes_bool {
4376   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4377 }{
4378   \excludecomment{nsproof}
4379 }
```

nexample

```
4380 \bool_if:NTF \c__mikoslides_notes_bool {
4381   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4382 }{
4383   \excludecomment{nexample}
4384 }
```

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

```
4385 \def\inputref@preskip{\smallskip}
4386 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
4387 \let\orig@inputref\inputref
4388 \def\inputref{\@ifstar\ninputref\orig@inputref}
4389 \newcommand\ninputref[2] [] {
4390   \bool_if:NT \c__mikoslides_notes_bool {
4391     \orig@inputref[#1]{#2}
4392   }
4393 }
```

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

32.3 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>}`.

```

4394 \newlength{\slidelogoheight}
4395
4396 \bool_if:NTF \c__mikosslides_notes_bool {
4397   \setlength{\slidelogoheight}{.4cm}
4398 }{
4399   \setlength{\slidelogoheight}{1cm}
4400 }
4401 \newsavebox{\slidelogo}
4402 \sbox{\slidelogo}{\TeX}
4403 \newrobustcmd{\setslidelogo}[1]{
4404   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4405 }
```

(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.

```

4406 \def\source{Michael Kohlhase}% customize locally
4407 \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.

```

4408 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4409 \newsavebox{\cclogo}
4410 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4411 \newif\ifcchref\cchreffalse
4412 \AtBeginDocument{
4413   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4414 }
4415 \def\licensing{
4416   \ifcchref
4417     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4418   \else
4419     {\usebox{\cclogo}}
4420   \fi
4421 }
4422 \newrobustcmd{\setlicensing}[2][]{
4423   \def\@url{#1}
4424   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4425   \ifx\@url\@empty
4426     \def\licensing{{\usebox{\cclogo}}}
4427   \else
4428     \def\licensing{
```

```

4429     \ifcchref
4430     \href{#1}{\usebox{\cclogo}}
4431     \else
4432     {\usebox{\cclogo}}
4433     \fi
4434   }
4435 \fi
4436 }

```

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

EdN:19 `\slidelabel` Now, we set up the slide label for the article mode.¹⁹

```

4437 \newrobustcmd\miko@slidelabel{
4438   \vbox to \slidelogoheight{
4439     \vss\hbox to \slidewidth
4440     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4441   }
4442 }

```

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

32.4 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.

```

4443 \def\Gin@mhrepos{}
4444 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4445 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4446 \newrobustcmd\frameimage[2][]{
4447   \stepcounter{slide}
4448   \bool_if:NT \c__mikoslides_frameimages_bool {
4449     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4450     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4451     \begin{center}
4452       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4453         \fbox{
4454           \ifx\Gin@ewidth\@empty
4455             \ifx\Gin@mhrepos\@empty
4456               \mhgraphics[width=\slidewidth,#1]{#2}
4457             \else
4458               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4459             \fi
4460           \else% Gin@ewidth empty
4461             \ifx\Gin@mhrepos\@empty
4462               \mhgraphics[#1]{#2}
4463             \else
4464               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4465             \fi
4466           \fi% Gin@ewidth empty
4467         }
4468       }{
4469         \ifx\Gin@ewidth\@empty

```

¹⁹EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

4470         \ifx\Gin@mhrepos\empty
4471           \mhgraphics[width=\slidewidth,#1]{#2}
4472         \else
4473           \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4474         \fi
4475         \ifx\Gin@mhrepos\empty
4476           \mhgraphics[#1]{#2}
4477         \else
4478           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4479         \fi
4480       \fi% Gin@ewidth empty
4481     }
4482   \end{center}
4483   \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4484   \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4485 }
4486 } % ifmks@sty@frameimages

```

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

32.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

4487 \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

```

4488 \AddToHook{begindocument}{
4489   \definecolor{green}{rgb}{0,.5,0}
4490   \definecolor{purple}{cmyk}{.3,1,0,.17}
4491 }

```

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`.

```

4492 % \def\STpresent#1{\textcolor{blue}{#1}}
4493 \def\defemph#1{\textcolor{magenta}{#1}}
4494 \def\symrefemph#1{\textcolor{cyan}{#1}}
4495 \def\compemph#1{\textcolor{blue}{#1}}
4496 \def\titleemph#1{\textcolor{blue}{#1}}
4497 \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.

```

4498 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4499 \def\smalltextwarning{
4500   \pgfuseimage{miko@small@dbend}
4501   \xspace
4502 }
4503 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4504 \newrobustcmd\textwarning{
4505   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4506   \xspace
4507 }
4508 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4509 \newrobustcmd\bigtextwarning{
4510   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4511   \xspace
4512 }

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

4513 \newrobustcmd\putgraphicsat[3]{
4514   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4515 }
4516 \newrobustcmd\putat[2]{
4517   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4518 }

```

32.6 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`.

```

4519 \bool_if:NT \c__mikoslides_sectocframes_bool {
4520   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4521     \newcounter{chapter}\counterwithin*{section}{chapter}
4522   }{
4523     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4524       \newcounter{chapter}\counterwithin*{section}{chapter}
4525     }
4526   }
4527 }

```

`\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
4528 \def\part@prefix{}
4529 \@ifpackageloaded{omdoc}{}{
4530   \str_case:VnF \__mikoslidestopsect {
4531     {part}{
4532       \int_set:Nn \l_document_structure_section_level_int {0}
4533       \def\thesection{\arabic{chapter}.\arabic{section}}
4534       \def\part@prefix{\arabic{chapter}.}
4535     }
4536     {chapter}{
4537       \int_set:Nn \l_document_structure_section_level_int {1}
4538       \def\thesection{\arabic{chapter}.\arabic{section}}
4539       \def\part@prefix{\arabic{chapter}.}
4540     }
4541   }{
4542     \int_set:Nn \l_document_structure_section_level_int {2}
4543     \def\part@prefix{}

```

```

4544 }
4545 }
4546
4547 \bool_if:NF \c__mikoslides_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 choses the L^AT_EX sectioning macros according to \section@level.

omgroup

```

4548 \renewenvironment{omgroup}[2][]{
4549   \__document_structure_omgroup_args:n { #1 }
4550   \int_incr:N \l_document_structure_omgroup_level_int
4551   \int_incr:N \l_document_structure_section_level_int
4552   \bool_if:NT \c__mikoslides_sectocframes_bool {
4553     \stepcounter{slide}
4554     \begin{frame}[noframenumbering]
4555     \vfill\Large\centering
4556     \red{
4557       \ifcase\l_document_structure_section_level_int\or
4558         \stepcounter{part}
4559         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4560         \def\currentsectionlevel{\omdoc@part@kw}
4561       \or
4562         \stepcounter{chapter}
4563         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4564         \def\currentsectionlevel{\omdoc@chapter@kw}
4565       \or
4566         \stepcounter{section}
4567         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4568         \def\currentsectionlevel{\omdoc@section@kw}
4569       \or
4570         \stepcounter{subsection}
4571         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4572         \def\currentsectionlevel{\omdoc@subsection@kw}
4573       \or
4574         \stepcounter{subsubsection}
4575         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4576         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4577       \or
4578         \stepcounter{mparagraph}
4579         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4580         \def\currentsectionlevel{\omdoc@paragraph@kw}
4581       \fi% end ifcase
4582       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4583       \quad #2%
4584     }%
4585     \vfill%
4586     \end{frame}%
4587   }
4588   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4589 }{}
4590 }

```

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

```

4591 \def\inserttheorembodyfont{\normalfont}
4592 \bool_if:NF \c__mikoslides_notes_bool {
4593   \defbeamertemplate{theorem begin}{miko}
4594   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4595     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4596     \inserttheorempunctuation\inserttheorembodyfont\hspace}
4597   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4598   \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.

```

4599   \expandafter\def\csname Parent2\endcsname{}
4600 }
4601 \bool_if:NT \c__mikoslides_notes_bool {
4602   \renewenvironment{columns}[1][]{%
4603     \par\noindent%
4604     \begin{minipage}%
4605       \slidewidth\centering\leavevmode%
4606   }{%
4607     \end{minipage}\par\noindent%
4608   }%
4609   \newsavebox\columnbox%
4610   \renewenvironment<>{column}[2][]{%
4611     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4612   }{%
4613     \end{minipage}\end{lrbox}\usebox\columnbox%
4614   }%
4615 }
4616 \bool_if:NTF \c__mikoslides_noproblems_bool {
4617   \newenvironment{problems}{}{}
4618 }{
4619   \excludecomment{problems}
4620 }
```

32.7 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.

```

4621 \gdef\printexcursions{}
4622 \newcommand\excursionref[2]{% label, text
4623   \bool_if:NT \c__mikoslides_notes_bool {
4624     \begin{omtext}[title=Excursion]
4625       #2 \sref[fallback=the appendix]{#1}.
4626     \end{omtext}
4627   }
4628 }
4629 \newcommand\activate@excursion[2][{}{
4630   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4631 }
4632 \newcommand\excursion[4][{}]{% repos, label, path, text
4633   \bool_if:NT \c__mikoslides_notes_bool {
4634     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4635   }
4636 }

```

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

\excursiongroup

```

4637 \keys_define:nn{mikoslides / excursiongroup }{
4638   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4639   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4640   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4641 }
4642 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4643   \tl_clear:N \l__mikoslides_excursion_intro_tl
4644   \str_clear:N \l__mikoslides_excursion_id_str
4645   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4646   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4647 }
4648 \newcommand\excursiongroup[1][{}]{
4649   \__mikoslides_excursion_args:n{ #1 }
4650   \ifdefempty\printexcursions{}% only if there are excursions
4651   {\begin{note}
4652     \begin{omgroup}[#1]{Excursions}%
4653     \ifdefempty\l__mikoslides_excursion_intro_tl{{
4654       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4655         \l__mikoslides_excursion_intro_tl
4656       }
4657     }
4658     \printexcursions%
4659     \end{omgroup}
4660   \end{note}}
4661 }
4662 \end{package}

```

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

Chapter 33

The Implementation

33.1 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.

```
4663 <*package>
4664 <@@=problems>
4665 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4666 \RequirePackage{l3keys2e,expl-keystr-compatible}
4667
4668 \keys_define:nn { problem / pkg }{
4669   notes      .default:n    = { true },
4670   notes      .bool_set:N   = \c__problems_notes_bool,
4671   gnotes     .default:n    = { true },
4672   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4673   hints      .default:n    = { true },
4674   hints      .bool_set:N   = \c__problems_hints_bool,
4675   solutions  .default:n    = { true },
4676   solutions  .bool_set:N   = \c__problems_solutions_bool,
4677   pts        .default:n    = { true },
4678   pts        .bool_set:N   = \c__problems_pts_bool,
4679   min        .default:n    = { true },
4680   min        .bool_set:N   = \c__problems_min_bool,
4681   boxed      .default:n    = { true },
4682   boxed      .bool_set:N   = \c__problems_boxed_bool,
4683   unknown    .code:n       = {}
4684 }
4685 \def\solutionstrue{
4686   \bool_set_true:N \c__problems_solutions_bool
4687 }
4688 \def\solutionsfalse{
4689   \bool_set_false:N \c__problems_solutions_bool
4690 }
4691
4692 \ProcessKeysOptions{ problem / pkg }
```

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


```

4693 \RequirePackage{stex-compatibility}
4694 \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.

```

4695 \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.

```

4696 \def\prob@problem@kw{Problem}
4697 \def\prob@solution@kw{Solution}
4698 \def\prob@hint@kw{Hint}
4699 \def\prob@note@kw{Note}
4700 \def\prob@gnote@kw{Grading}
4701 \def\prob@pt@kw{pt}
4702 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4703 \@ifpackageloaded{babel}{
4704   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4705   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4706     \input{problem-ngerman.ldf}
4707   }
4708   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4709     \input{problem-finnish.ldf}
4710   }
4711   \clist_if_in:NnT \l_tmpa_clist {french}{
4712     \input{problem-french.ldf}
4713   }
4714   \clist_if_in:NnT \l_tmpa_clist {russian}{
4715     \input{problem-russian.ldf}
4716   }
4717 }{}

```

33.2 Problems and Solutions

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

```

4718 \keys_define:nn{ problem / problem }{
4719   id      .str_set:x:N = \l__problems_prob_id_str,
4720   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4721   min     .tl_set:N    = \l__problems_prob_min_tl,
4722   title   .tl_set:N    = \l__problems_prob_title_tl,
4723   refnum  .int_set:N   = \l__problems_prob_refnum_int
4724 }
4725 \cs_new_protected:Nn \__problems_prob_args:n {
4726   \str_clear:N \l__problems_prob_id_str
4727   \tl_clear:N \l__problems_prob_pts_tl
4728   \tl_clear:N \l__problems_prob_min_tl
4729   \tl_clear:N \l__problems_prob_title_tl

```

```

4730 \int_zero_new:N \l__problems_prob_refnum_int
4731 \keys_set:nn { problem / problem }{ #1 }
4732 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4733   \let\l__problems_inclprob_refnum_int\undefined
4734 }
4735 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4736 \newcounter{problem}
4737 \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.

```

4738 \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

```

4739 \newcommand\prob@number{
4740   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4741     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4742   }{
4743     \int_if_exist:NTF \l__problems_prob_refnum_int {
4744       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4745     }{
4746       \prob@label\theproblem
4747     }
4748   }
4749 }

```

(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.

```

4750 \newcommand\prob@title[3]{%
4751   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4752     #2 \l__problems_inclprob_title_tl #3
4753   }{
4754     \tl_if_exist:NTF \l__problems_prob_title_tl {
4755       #2 \l__problems_prob_title_tl #3
4756     }{
4757       #1
4758     }
4759   }
4760 }

```

(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.

```

4761 \def\prob@heading{
4762   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4763   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4764 }

```

(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.

`problem`

```

4765 \newenvironment{problem}[1][1]{
4766   \__problems_prob_args:n{#1}%\sref@target%
4767   \@in@omtexttrue% we are in a statement (for inline definitions)
4768   \stepcounter{problem}\record@problem
4769   \def\current@section@level{\prob@problem@kw}
4770   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4771 }%
4772   {\smallskip}
4773   \bool_if:NT \c__problems_boxed_bool {
4774     \surroundwithmdframed{problem}
4775   }

```

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

```

4776 \def\record@problem{
4777   \protected@write\@auxout{}
4778   {
4779     \string\@problem{\prob@number}
4780     {
4781       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4782         \l__problems_inclprob_pts_tl
4783       }{
4784         \l__problems_prob_pts_tl
4785       }
4786     }%
4787     {
4788       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4789         \l__problems_inclprob_min_tl
4790       }{
4791         \l__problems_prob_min_tl
4792       }
4793     }
4794   }
4795 }

```

(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).

```

4796 \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.

```

4797 \keys_define:nn { problem / solution }{
4798   id          .str_set_x:N = \l__problems_solution_id_str ,
4799   for         .tl_set:N    = \l__problems_solution_for_tl ,
4800   height      .dim_set:N   = \l__problems_solution_height_dim ,
4801   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4802   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4803   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4804 }
4805 \cs_new_protected:Nn \__problems_solution_args:n {
4806   \str_clear:N \l__problems_solution_id_str
4807   \tl_clear:N \l__problems_solution_for_tl
4808   \tl_clear:N \l__problems_solution_srccite_tl
4809   \clist_clear:N \l__problems_solution_creators_clist
4810   \clist_clear:N \l__problems_solution_contributors_clist
4811   \dim_zero:N \l__problems_solution_height_dim
4812   \keys_set:nn { problem / solution }{ #1 }
4813 }

```

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

```

4814 \newcommand\@startsolution[1][ ]{
4815   \__problems_solution_args:n { #1 }
4816   \@in@omtexttrue% we are in a statement.
4817   \bool_if:NF \c__problems_boxed_bool { \hrule }
4818   \smallskip\noindent
4819   {\textbf\prob@solution@kw : \enspace}
4820   \begin{small}
4821   \def\current@section@level{\prob@solution@kw}
4822   \ignorespacesandpars
4823 }

```

\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.

```

4824 \newcommand\startsolutions{
4825   \specialcomment{solution}{\@startsolution}{
4826     \bool_if:NF \c__problems_boxed_bool {
4827       \hrule\medskip
4828     }
4829     \end{small}%
4830   }
4831   \bool_if:NT \c__problems_boxed_bool {
4832     \surroundwithmdframed{solution}
4833   }
4834 }

```

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

\stopsolutions

```

4835 \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.

```

4836 \bool_if:NTF \c_problems_solutions_bool {
4837   \startsolutions
4838 }{
4839   \stopsolutions
4840 }

```

exnote

```

4841 \bool_if:NTF \c_problems_notes_bool {
4842   \newenvironment{exnote}[1][]{
4843     \par\smallskip\hrule\smallskip
4844     \noindent\textbf{\prob@note@kw : }\small
4845   }{
4846     \smallskip\hrule
4847   }
4848 }{
4849   \excludecomment{exnote}
4850 }

```

hint

```

4851 \bool_if:NTF \c_problems_notes_bool {
4852   \newenvironment{hint}[1][]{
4853     \par\smallskip\hrule\smallskip
4854     \noindent\textbf{\prob@hint@kw :~ }\small
4855   }{
4856     \smallskip\hrule
4857   }
4858   \newenvironment{exhint}[1][]{
4859     \par\smallskip\hrule\smallskip
4860     \noindent\textbf{\prob@hint@kw :~ }\small
4861   }{
4862     \smallskip\hrule
4863   }
4864 }{
4865   \excludecomment{hint}
4866   \excludecomment{exhint}
4867 }

```

gnote

```

4868 \bool_if:NTF \c_problems_notes_bool {
4869   \newenvironment{gnote}[1][]{
4870     \par\smallskip\hrule\smallskip
4871     \noindent\textbf{\prob@gnote@kw : }\small
4872   }{
4873     \smallskip\hrule
4874   }
4875 }{
4876   \excludecomment{gnote}
4877 }

```

33.3 Multiple Choice Blocks

```

4878 \newenvironment{mcb}{
4879   \begin{enumerate}
4880 }{
4881   \end{enumerate}
4882 }

```

we define the keys for the mcc macro

```

4883 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4884   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4885     \bool_set_true:N #1
4886   }{
4887     \bool_set_false:N #1
4888   }
4889 }
4890 \keys_define:nn { problem / mcc }{
4891   id          .str_set:x:N = \l__problems_mcc_id_str ,
4892   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4893   T           .default:n    = { true } ,
4894   T           .bool_set:N    = \l__problems_mcc_t_bool ,
4895   F           .default:n    = { true } ,
4896   F           .bool_set:N    = \l__problems_mcc_f_bool ,
4897   Ttext       .code:n       = {
4898     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4899   } ,
4900   Ftext       .code:n       = {
4901     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4902   }
4903 }
4904 \cs_new_protected:Nn \l__problems_mcc_args:n {
4905   \str_clear:N \l__problems_mcc_id_str
4906   \tl_clear:N \l__problems_mcc_feedback_tl
4907   \bool_set_true:N \l__problems_mcc_t_bool
4908   \bool_set_true:N \l__problems_mcc_f_bool
4909   \bool_set_true:N \l__problems_mcc_Ttext_bool
4910   \bool_set_false:N \l__problems_mcc_Ftext_bool
4911   \keys_set:nn { problem / mcc }{ #1 }
4912 }

```

\mcc

```

4913 \newcommand\mcc[2][] {
4914   \l__problems_mcc_args:n{ #1 }
4915   \item #2
4916   \bool_if:NT \c__problems_solutions_bool {
4917     \
4918     \bool_if:NT \l__problems_mcc_t_bool {
4919       % TODO!
4920       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
4921     }
4922     \bool_if:NT \l__problems_mcc_f_bool {

```

²⁰EdNOTE: MK: maybe import something better here from a dedicated MC package

```

4923      % TODO!
4924      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4925    }
4926    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4927      !
4928    }{
4929      \l__problems_mcc_feedback_tl
4930    }
4931  }
4932 } %solutions

```

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

33.4 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.

```

4933
4934 \keys_define:nn{ problem / inclproblem }{
4935   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4936   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4937   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4938   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4939   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4940   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4941 }
4942 \cs_new_protected:Nn \__problems_inclprob_args:n {
4943   % \str_clear:N \l__problems_prob_id_str
4944   \tl_clear:N \l__problems_inclprob_pts_tl
4945   \tl_clear:N \l__problems_inclprob_min_tl
4946   \tl_clear:N \l__problems_inclprob_title_tl
4947   \int_zero_new:N \l__problems_inclprob_refnum_int
4948   \str_clear:N \l__problems_inclprob_mhrepos_str
4949   \keys_set:nn { problem / inclproblem }{ #1 }
4950   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4951     \let\l__problems_inclprob_pts_tl\undefined
4952   }
4953   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4954     \let\l__problems_inclprob_min_tl\undefined
4955   }
4956   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4957     \let\l__problems_inclprob_title_tl\undefined
4958   }
4959   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4960     \let\l__problems_inclprob_refnum_int\undefined
4961   }
4962 }
4963
4964 \cs_new_protected:Nn \__problems_inclprob_clear: {
4965   % \str_clear:N \l__problems_prob_id_str
4966   \let\l__problems_inclprob_pts_tl\undefined
4967   \let\l__problems_inclprob_min_tl\undefined

```

```

4968 \let\l__problems_inclprob_title_tl\undefined
4969 \let\l__problems_inclprob_refnum_int\undefined
4970 \let\l__problems_inclprob_mhrepos_str\undefined
4971 }
4972
4973 \newcommand\includeproblem[2][ ]{
4974   \__problems_inclprob_args:n{ #1 }
4975   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4976     \input{#2}
4977   }{
4978     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4979       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4980     }
4981   }
4982   \__problems_inclprob_clear:
4983 }

```

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

33.5 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.

```

4984 \AddToHook{enddocument}{
4985   \bool_if:NT \c__problems_pts_bool {
4986     \message{Total:~\arabic{pts}~points}
4987   }
4988   \bool_if:NT \c__problems_min_bool {
4989     \message{Total:~\arabic{min}~minutes}
4990   }
4991 }

```

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

```

4992 \def\pts#1{
4993   \bool_if:NT \c__problems_pts_bool {
4994     \marginpar{#1~\prob@pt@kw}
4995   }
4996 }
4997 \def\min#1{
4998   \bool_if:NT \c__problems_min_bool {
4999     \marginpar{#1~\prob@min@kw}
5000   }
5001 }

```

\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.

```

5002 \newcounter{pts}
5003 \def\show@pts{
5004   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5005     \bool_if:NT \c__problems_pts_bool {
5006       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5007       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```



```

5008     }
5009   }{
5010     \tl_if_exist:NT \l__problems_prob_pts_tl {
5011       \bool_if:NT \c__problems_pts_bool {
5012         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5013         \addtocounter{pts}{\l__problems_prob_pts_tl}
5014       }
5015     }
5016   }
5017 }

```

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

\show@min

```

5018 \newcounter{min}
5019 \def\show@min{
5020   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5021     \bool_if:NT \c__problems_min_bool {
5022       \marginpar{\l__problems_inclprob_pts_tl;min}
5023       \addtocounter{min}{\l__problems_inclprob_min_tl}
5024     }
5025   }{
5026     \tl_if_exist:NT \l__problems_prob_min_tl {
5027       \bool_if:NT \c__problems_min_bool {
5028         \marginpar{\l__problems_prob_min_tl;min}
5029         \addtocounter{min}{\l__problems_prob_min_tl}
5030       }
5031     }
5032   }
5033 }
5034 \</package>

```

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

Chapter 34

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.

34.1 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.

```
5035 <@@=hwexam>
5036 <*cls>
5037 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5038 \RequirePackage{l3keys2e,expl-keystr-compatible}
5039 \DeclareOption*{
5040   \PassOptionsToClass{\CurrentOption}{omdoc}
5041   \PassOptionsToPackage{\CurrentOption}{stex}
5042   \PassOptionsToPackage{\CurrentOption}{hwexam}
5043   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5044 }
5045 \ProcessOptions
```

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

```
5046 \LoadClass{omdoc}
5047 \RequirePackage{stex}
5048 \RequirePackage{hwexam}
5049 \RequirePackage{tikzinput}
5050 \RequirePackage{graphicx}
5051 \RequirePackage{a4wide}
5052 \RequirePackage{amssymb}
5053 \RequirePackage{amstext}
5054 \RequirePackage{amsmath}
```

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

```

5055 \newcommand\assig@default@type{\hwexam@assignment@kw}
5056 \def\document@hwexamtype{\assig@default@type}
5057 <@@=document_structure>
5058 \keys_define:nn { document-structure / document }{
5059 id .str_set_x:N = \c_document_structure_document_id_str,
5060 hwexamtype .tl_set:N = \document@hwexamtype
5061 }
5062 <@@=hwexam>
5063 </cls>

```

Chapter 35

Implementation: The hwexam Package

35.1 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.

```
5064 \*package>
5065 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5066 \RequirePackage{l3keys2e,expl-keystr-compat}
5067
5068 \newif\iftest\testfalse
5069 \DeclareOption{test}{\testtrue}
5070 \newif\ifmultiple\multiplefalse
5071 \DeclareOption{multiple}{\multipletrue}
5072 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5073 \ProcessOptions
```

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

```
5074 \RequirePackage{keyval}[1997/11/10]
5075 \RequirePackage{problem}
```

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

```
5076 \newcommand\hwexam@assignment@kw{Assignment}
5077 \newcommand\hwexam@given@kw{Given}
5078 \newcommand\hwexam@due@kw{Due}
5079 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5080 space}%
5081 \newcommand\correction@probs@kw{prob.}%
5082 \newcommand\correction@pts@kw{total}%
5083 \newcommand\correction@reached@kw{reached}%
5084 \newcommand\correction@sum@kw{Sum}%
5085 \newcommand\correction@grade@kw{grade}%
5086 \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

```

5087 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5088
5089 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5090 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5091   \input{hwexam-ngerman.ldf}
5092 }
5093 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5094   \input{hwexam-finnish.ldf}
5095 }
5096 \clist_if_in:NnT \l_tmpa_clist {french}{
5097   \input{hwexam-french.ldf}
5098 }
5099 \clist_if_in:NnT \l_tmpa_clist {russian}{
5100   \input{hwexam-russian.ldf}
5101 }

```

35.2 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.

```

5102 \newcounter{assignment}
5103 \numberproblemsin{assignment}
5104 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5105 \keys_define:nn { hwexam / assignment } {
5106   id .str_set:N = \l__hwexam_assign_id_str,
5107   number .int_set:N = \l__hwexam_assign_number_int,
5108   title .tl_set:N = \l__hwexam_assign_title_tl,
5109   type .tl_set:N = \l__hwexam_assign_type_tl,
5110   given .tl_set:N = \l__hwexam_assign_given_tl,
5111   due .tl_set:N = \l__hwexam_assign_due_tl,
5112   loadmodules .code:n = {
5113     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5114   }
5115 }
5116 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5117   \str_clear:N \l__hwexam_assign_id_str
5118   \int_set:Nn \l__hwexam_assign_number_int {-1}
5119   \tl_clear:N \l__hwexam_assign_title_tl
5120   \tl_clear:N \l__hwexam_assign_type_tl
5121   \tl_clear:N \l__hwexam_assign_given_tl
5122   \tl_clear:N \l__hwexam_assign_due_tl
5123   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5124   \keys_set:nn { hwexam / assignment }{ #1 }
5125 }

```

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.

```

5126 \newcommand\given@due[2]{
5127 \bool_lazy_all:nF {
5128 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5129 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5130 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5131 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5132 }{ #1 }
5133
5134 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5135 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5136 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5137 }
5138 }{
5139 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5140 }
5141
5142 \bool_lazy_or:nnF {
5143 \bool_lazy_and_p:nn {
5144 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5145 }{
5146 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5147 }
5148 }{
5149 \bool_lazy_and_p:nn {
5150 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5151 }{
5152 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5153 }
5154 }{ ,~ }
5155
5156 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5157 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5158 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5159 }
5160 }{
5161 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5162 }
5163
5164 \bool_lazy_all:nF {
5165 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5166 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5167 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5168 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5169 }{ #2 }
5170 }

```

`\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.

```

5171 \newcommand\assignment@title[3]{

```

```

5172 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5173 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5174 #1
5175 }{
5176 #2\l__hwexam_assign_title_tl#3
5177 }
5178 }{
5179 #2\l__hwexam_inclassassign_title_tl#3
5180 }
5181 }

```

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

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

```

5182 \newcommand\assignment@number{
5183 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5184 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5185 \int_use:N \l__hwexam_assign_number_int
5186 }
5187 }{
5188 \int_use:N \l__hwexam_inclassassign_number_int
5189 }
5190 }

```

(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.

```

5191 \newenvironment{assignment}[1][]{
5192 \__hwexam_assignment_args:n { #1 }
5193 %\sref@target
5194 \let\__hwexamnum\l__hwexam_assign_number_int
5195 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5196 \stepcounter{assignment}
5197 }{
5198 \setcounter{assignment}{\int_use:N\__hwexamnum}
5199 }
5200 \setcounter{problem}{0}
5201 \def\current@section@level{\document@hwexamtype}
5202 %\sref@label@id{\document@hwexamtype \thesection}
5203 \begin{@assignment}
5204 }{
5205 \end{@assignment}
5206 }

```

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

```

5207 \def\__hwexasstitle{
5208 \protect\document@hwexamtype~\arabic{assignment}
5209 \assignment@title{}\;{} \; -- \given@due{}\}
5210 }

```

```

5211 \ifmultiple
5212 \newenvironment{@assignment}{
5213 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5214 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5215 }{
5216 \begin{omgroup}{\__hwexasstitle}
5217 }
5218 }{
5219 \end{omgroup}
5220 }

```

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

```

5221 \else
5222 \newenvironment{@assignment}{
5223 \begin{center}\bf
5224 \Large\@title\strut\
5225 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}\}
5226 \large\given@due{--\;}{\;}{--}
5227 \end{center}
5228 }{}
5229 \fi% multiple

```

35.3 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.

```

5230 \keys_define:nn { hwexam / inclassignment } {
5231 %id .str_set_x:N = \l__hwexam_assign_id_str,
5232 number .int_set:N = \l__hwexam_inclassign_number_int,
5233 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5234 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5235 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5236 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5237 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5238 }
5239 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5240 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5241 \tl_clear:N \l__hwexam_inclassign_title_tl
5242 \tl_clear:N \l__hwexam_inclassign_type_tl
5243 \tl_clear:N \l__hwexam_inclassign_given_tl
5244 \tl_clear:N \l__hwexam_inclassign_due_tl
5245 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5246 \keys_set:nn { hwexam / inclassignment }{ #1 }
5247 }
5248 \__hwexam_inclassignment_args:n {}
5249
5250 \newcommand\inputassignment[2][ ]{
5251 \__hwexam_inclassignment_args:n { #1 }
5252 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5253 \input{#2}
5254 }{
5255 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```



```

5256 \input{\mhp{path}\l__hwexam_inclasssign_mhrepos_str}{#2}}
5257 }
5258 }
5259 \__hwexam_inclasssign_args:n {}
5260 }
5261 \newcommand\includeassignment[2][ ]{
5262 \newpage
5263 \inputassignment[#1]{#2}
5264 }

```

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

35.4 Typesetting Exams

\quizheading

```

5265 \ExplSyntaxOff
5266 \newcommand\quizheading[1]{%
5267 \def\@tas{#1}%
5268 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5269 \ifx\@tas\empty\else%
5270 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5271 \fi%
5272 }
5273 \ExplSyntaxOn

```

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

\testheading

```

5274 \keys_define:nn { hwexam / testheading } {
5275 min .tl_set:N = \l__hwexam_testheading_min_tl,
5276 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5277 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5278 }
5279 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5280 \tl_clear:N \l__hwexam_testheading_min_tl
5281 \tl_clear:N \l__hwexam_testheading_duration_tl
5282 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5283 \keys_set:nn { hwexam / testheading }{ #1 }
5284 }
5285 \newenvironment{testheading}[1][ ]{
5286 \__hwexam_testheading_args:n{ #1 }
5287 \noindent\large{Name:~\hfill
5288 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5289 \begin{center}
5290 \Large\textbf{\@title}\[1ex]
5291 \large\@date\[3ex]
5292 \end{center}
5293 \textbf{You~have~
5294 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5295 \l__hwexam_testheading_min_tl~minutes
5296 }{
5297 \l__hwexam_testheading_duration_tl
5298 }~

```

```

5299 (sharp)~for~the~test
5300 };\
5301 Write~the~solutions~to~the~sheet.
5302 \par\noindent
5303 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5304 \advance\check@time by -\theassignment@totalmin
5305 The~estimated~time~for~solving~this~exam~is~
5306 {\theassignment@totalmin}-minutes,~
5307 leaving~you~{\the\check@time}-minutes~for~revising~
5308 your~exam.
5309
5310 \par\noindent
5311 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5312 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5313 You~can~reach~{\theassignment@totalpts}-points~if~you~
5314 solve~all~problems.~You~will~only~need~
5315 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5316 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5317 \vfill
5318 \begin{center}
5319 {
5320 \Large\em You~have~ample~time,~so~take~it~slow~
5321 and~avoid~rushing~to~mistakes!\}[2ex]
5322 Different~problems~test~different~skills~and~
5323 knowledge,~so~do~not~get~stuck~on~one~problem.
5324 }
5325 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5326 \end{center}
5327 }{
5328 \newpage
5329 }

```

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

\testspace

```

5330 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5331 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5332 \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.

```

5333 <@=problems>
5334 \renewcommand\@problem[3]{
5335 \stepcounter{assignment@probs}
5336 \def\__problemspts{#2}

```

```

5337 \ifx\__problemspts\@empty\else
5338 \addtocounter{assignment@totalpts}{#2}
5339 \fi
5340 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5341 \xdef\correction@probs{\correction@probs & #1}%
5342 \xdef\correction@pts{\correction@pts & #2}
5343 \xdef\correction@reached{\correction@reached & }
5344 }
5345 \<@=hwexam>

```

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

\correction@table This macro generates the correction table

```

5346 \newcounter{assignment@probs}
5347 \newcounter{assignment@totalpts}
5348 \newcounter{assignment@totalmin}
5349 \def\correction@probs{\correction@probs@kw}%
5350 \def\correction@pts{\correction@pts@kw}%
5351 \def\correction@reached{\correction@reached@kw}%
5352 \def\after@correction@table{}%
5353 \stepcounter{assignment@probs}
5354 \newcommand\correction@table{
5355 \resizebox{\textwidth}{!}{%
5356 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5357 &\multicolumn{\theassignment@probs}{c|}|%|
5358 {\footnotesize\correction@forgrading@kw} &\\ \hline
5359 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5360 \correction@pts & \theassignment@totalpts & \\ \hline
5361 \correction@reached & & \[.7cm]\hline
5362 \end{tabular}}
5363 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5364 \end{package}

```

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

35.5 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\bierrfont=../assignments/bierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\bierrglas{{\bierrfont\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{\bierrglas}

```