

The \TeX 3 Package *

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

2022-01-25

Abstract

TODO

*Version 3.0 (last revised 2022-01-25)

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	135
26.2	Assertions	138
26.3	Examples	140
26.4	Logical Paragraphs	142
27	The Implementation	145
27.1	Package Options	145
27.2	Proofs	145
27.3	Justifications	151
28	STEX-Others Implementation	153
29	STEX-Metatheory Implementation	154
30	Tikzinput Implementation	157
31	document-structure.sty Implementation	159
31.1	The OMDoc Class	159
31.2	Class Options	159
31.3	Beefing up the document environment	160
31.4	Implementation: OMDoc Package	160
31.5	Package Options	160
31.6	Document Structure	162
31.7	Front and Backmatter	165
31.8	Global Variables	167
32	MiKoSlides – Implementation	168
32.1	Class and Package Options	168
32.2	Notes and Slides	170
32.3	Header and Footer Lines	174
32.4	Frame Images	175
32.5	Colors and Highlighting	176
32.6	Sectioning	177
32.7	Excursions	179

33 The Implementation	181
33.1 Package Options	181
33.2 Problems and Solutions	182
33.3 Multiple Choice Blocks	187
33.4 Including Problems	188
33.5 Reporting Metadata	189
34 Implementation: The hwexam Class	191
34.1 Class Options	191
35 Implementation: The hwexam Package	193
35.1 Package Options	193
35.2 Assignments	194
35.3 Including Assignments	197
35.4 Typesetting Exams	198
35.5 Leftovers	200

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  $\textcolor{teal}{\$}\textcolor{teal}{\textit{add}}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{\textit{add}}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

The operator $\textcolor{teal}{+}$ adds two elements, as in $a\textcolor{teal}{+}b$.

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

Example 7

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

$\textcolor{teal}{\textit{Multiplication}}$ (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:NTF</code>	\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 <code>.sms</code> -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_str`

All information of a module is stored as a property list. `\l_stex_current_module_str` 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_str` 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:cn {c_stex_module_\l_stex_current_module_str_prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str_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:cn {c_stex_module_\l_stex_current_module_str _prop} { ns }?
\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { name }\\
Language:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { lang }\\
Signature:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} { sig }\\
Metatheory:-\prop_item:cn {c_stex_module_\l_stex_current_module_str _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?collect, 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 } }
 $\bar{foobar} a\{b,c,d,e,f\}g$  and  $\bar{foobar}[foo] a\{b,c\}g$  and  $\bar{foobar} abc$ 

\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  $\mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$ 
 $\displaystyle \plus{a,\mult{b,c}}$  and  $\displaystyle \mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$ 
\withbrackets[ {  $\displaystyle \plus{a,\mult{b,c}}$  } and  $\displaystyle \mult{a,\plus{\frac{ab}{b},\frac{ac}{c}}}$  }
\end{module}

```

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

TeX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

`mathstructure` TODO

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.

`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 spfcase environments that mark up the cases one by one.
spfcase	The content of a pfcases environment are a sequence of case proofs marked up in the pfcase 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 pfcase environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcase 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 .

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

This package supplies an infrastructure for writing OMDOC documents in $\text{\texttt{LaTeX}}$. This includes a simple structure sharing mechanism for $\text{\texttt{sTeX}}$ 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 $\text{\texttt{sTeX}}$ sources, or after translation.

14.1 Introduction

$\text{\texttt{sTeX}}$ is a version of $\text{\texttt{TeX/LaTeX}}$ that allows to markup $\text{\texttt{TeX/LaTeX}}$ documents semantically without leaving the document format, essentially turning $\text{\texttt{TeX/LaTeX}}$ 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 $\text{\texttt{sTeX}}$ sources, or after translation. Currently, trans-document referencing provided by this package can only be used in the $\text{\texttt{sTeX}}$ collection.

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

source and the formatter does the copying during document formatting/presentation.⁶

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` 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 L^AT_EXML transformation.

`omgroup` The structure of the document is given by the `omgroup` environment just like in OM-DOC. In the L^AT_EX 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.barderv,loadmodules]{Introducing $\protect\bar$ Derivations}
```

`blindomgroup` L^AT_EX 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 \LaTeX and OMDoc. To support semantic course notes, it extends the notion of mixing frames and explanatory text, but rather than treating the frames as images (or integrating their contents into the flowing text), the `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, the \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{<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 — 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` 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.

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

ST_EX -Basics Implementation

18.1 The ST_EXDocument Class

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

```
1 <*cls>
2
3 %%%%%%%%% basics.dtx %%%%%%%%%
4
5 \RequirePackage{expl3,l3keys2e}
6 \ProvidesExplClass{stex}{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
23 %\RequirePackage{morewrites}
24 %\RequirePackage{amsmath}
25
```

Package options:

```

26 \keys_define:nn { stex } {
27   debug      .clist_set:N = \c_stex_debug_clist ,
28   showmods   .bool_set:N = \c_stex_showmods_bool ,
29   lang        .clist_set:N = \c_stex_languages_clist ,
30   mathhub     .tl_set_x:N = \mathhub ,
31   sms         .bool_set:N = \c_stex_persist_mode_bool ,
32   image       .bool_set:N = \c_tikzinput_image_bool ,
33   unknown     .code:n      = {}
34 }
35 \ProcessKeysOptions { stex }

```

\stex The sTeX logo:

\sTeX

```

36 \protected\def\stex{%
37   \@ifundefined{texorpdfstring}%
38   {\let\texorpdfstring\@firstoftwo}%
39   }%
40   \texorpdfstring{\raisebox{-.5ex}{S}\kern-.5ex\TeX}{sTeX}\xspace%
41 }
42 \def\sTeX{\stex}

```

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

18.3 Messages and logging

```

43 <@@=stex_log>

Warnings and error messages
44 \msg_new:nnn{stex}{error/unknownlanguage}{
45   Unknown~language:~#1
46 }
47 \msg_new:nnn{stex}{warning/nomathhub}{
48   MATHHUB~system~variable~not~found~and~no~
49   \detokenize{\mathhub}~value~set!
50 }
51 \msg_new:nnn{stex}{error/deactivated-macro}{
52   The~\detokenize{#1}~command~is~only~allowed~in~#2!
53 }

```

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

```

54 \cs_new_protected:Nn \stex_debug:nn {
55   \clist_if_in:NnTF \c_stex_debug_clist { all } {
56     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
57       \\Debug~#1:~#2\\
58     }
59     \msg_none:nn{stex}{debug / #1}
60   }{
61     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
62       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
63         \\Debug~#1:~#2\\
64       }
65       \msg_none:nn{stex}{debug / #1}
66     }
67   }
68 }

```

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

Redirecting messages:

```

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

```

18.4 Persistence

78 `<@@=stex_persist>`

`\c__stex_persist_sms_iow` File variable used for the sms-File

```

79 \iow_new:N \c__stex_persist_sms_iow
80 \AddToHook{begindocument}{
81   \bool_if:NTF \c_stex_persist_mode_bool {
82     \ExplSyntaxOn \input{\jobname.sms} \ExplSyntaxOff
83   } {
84     \iow_open:Nn \c__stex_persist_sms_iow {\jobname.sms}
85   }
86 }
87 \AddToHook{enddocument}{
88   \bool_if:NF \c_stex_persist_mode_bool {
89     \iow_close:N \c__stex_persist_sms_iow
90   }
91 }

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

92 \cs_new_protected:Nn \stex_add_to_sms:n {
93   \bool_if:NF \c_stex_persist_mode_bool {
94     \iow_now:Nn \c__stex_persist_sms_iow { #1 }
95   }
96 }

```

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

18.5 HTML Annotations

97 `<@@=stex_annotate>`
98 `\RequirePackage{rustex}`

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

```

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

```

`\if@latexml` Conditionals for L^AT_EX_ML:
`\latexml_if_p:`
`\latexml_if:TF`

```

100 \ifcsname if@latexml\endcsname\else

```

```

101 \expandafter\newif\csname if@latexml\endcsname\@latexmlfalse
102 \fi
103
104 \prg_new_conditional:Nnn \latexml_if: {p, T, F, TF} {
105   \if@latexml
106     \prg_return_true:
107   \else:
108     \prg_return_false:
109   \fi:
110 }

```

(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`

```

111 \tl_new:N \l__stex_annotate_arg_tl
112 \tl_const:Nx \c__stex_annotate_emptyarg_tl {
113   \rustex_if:TF {
114     \rustex_direct_HTML:n { \c_ampersand_str lrm; }
115   }{-}
116 }

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

`_stex_annotate_checkempty:n`

```

117 \cs_new_protected:Nn \_stex_annotate_checkempty:n {
118   \tl_set:Nn \l__stex_annotate_arg_tl { #1 }
119   \tl_if_empty:NT \l__stex_annotate_arg_tl {
120     \tl_set_eq:NN \l__stex_annotate_arg_tl \c__stex_annotate_emptyarg_tl
121   }
122 }

```

(End definition for _stex_annotate_checkempty:n.)

`\l_stex_html_do_output_bool` Whether to (locally) produce HTML output

```

\stex_if_do_html:
123 \bool_new:N \l_stex_html_do_output_bool
124 \bool_set_true:N \l_stex_html_do_output_bool
125 \prg_new_conditional:Nnn \stex_if_do_html: {p,T,F,TF} {
126   \bool_if:nTF \l_stex_html_do_output_bool
127     \prg_return_true: \prg_return_false:
128 }

```

(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

```

129 \cs_new_protected:Nn \stex_suppress_html:n {
130   \exp_args:Nne \use:nn {
131     \bool_set_false:N \l_stex_html_do_output_bool
132     #1
133   }{
134     \stex_if_do_html:T {
135       \bool_set_true:N \l_stex_html_do_output_bool
136     }
137   }
138 }

```

(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.

```

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

```

```

187   }{
188     \par\rustex_annotate_HTML_end:
189   }
190 }{
191   \latexml_if:TF {
192     \cs_new_protected:Nn \stex_annotate:nnn {
193       \__stex_annotate_checkempty:n { #3 }
194       \mode_if_math:TF {
195         \cs:w latexml@annotate@math\cs_end:{#1}{#2}{
196           \tl_use:N \l__stex_annotate_arg_tl
197         }
198       }{
199         \cs:w latexml@annotate@text\cs_end:{#1}{#2}{
200           \tl_use:N \l__stex_annotate_arg_tl
201         }
202       }
203     }
204     \cs_new_protected:Nn \stex_annotate_invisible:n {
205       \__stex_annotate_checkempty:n { #1 }
206       \mode_if_math:TF {
207         \cs:w latexml@invisible@math\cs_end:{
208           \tl_use:N \l__stex_annotate_arg_tl
209         }
210       } {
211         \cs:w latexml@invisible@text\cs_end:{
212           \tl_use:N \l__stex_annotate_arg_tl
213         }
214       }
215     }
216     \cs_new_protected:Nn \stex_annotate_invisible:nnn {
217       \__stex_annotate_checkempty:n { #3 }
218       \cs:w latexml@annotate@invisible\cs_end:{#1}{#2}{
219         \tl_use:N \l__stex_annotate_arg_tl
220       }
221     }
222     \NewDocumentEnvironment{stex_annotate_env} { m m } {
223       \par\begin{latexml@annotateenv}{#1}{#2}
224     }{
225       \par\end{latexml@annotateenv}
226     }
227   }{
228     \cs_new_protected:Nn \stex_annotate:nnn {#3}
229     \cs_new_protected:Nn \stex_annotate_invisible:n {}
230     \cs_new_protected:Nn \stex_annotate_invisible:nnn {}
231     \NewDocumentEnvironment{stex_annotate_env} { m m } {}{}
232   }
233 }

```

(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

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

235 \prop_const_from_keyval:Nn \c_stex_languages_prop {
236   en = english ,
237   de = ngerman ,
238   ar = arabic ,
239   bg = bulgarian ,
240   ru = russian ,
241   fi = finnish ,
242   ro = romanian ,
243   tr = turkish ,
244   fr = french
245 }
246
247 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
248   english   = en ,
249   ngerman   = de ,
250   arabic    = ar ,
251   bulgarian = bg ,
252   russian   = ru ,
253   finnish   = fi ,
254   romanian  = ro ,
255   turkish   = tr ,
256   french    = fr
257 }
258 % todo: chinese simplified (zhs)
259 %       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:

```

260 \clist_if_empty:NF \c_stex_languages_clist {
261   \clist_clear:N \l_tmpa_clist
262   \clist_map_inline:Nn \c_stex_languages_clist {
263     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
264       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
265     } {
266       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
267     }
268   }
269   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
270   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
271 }

```

18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

272 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
273   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
274   \def#1{
275     \msg_error:nxxx{stex}{error/deactivated-macro}{#1}{#2}
276   }
277 }

```


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

\stex_reactivate_macro:N

```
278 \cs_new_protected:Nn \stex_reactivate_macro:N {  
279   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname  
280 }
```

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

```
281 \</package>
```

Chapter 19

STEX -MathHub Implementation

```
282 <*package>
283
284 %%%%%%%%%% mathhub.dtx %%%%%%%%%%
285
286 <@@=stex_path>
287
288 Warnings and error messages
289 \msg_new:nnn{stex}{error/norepository}{
290   No~archive~#1~found~in~#2
291 }
292 \msg_new:nnn{stex}{error/notinarchive}{
293   Not~currently~in~an~archive,~but~\detokenize{#1}~
294   needs~one!
295 }
296 \msg_new:nnn{stex}{error/nofile}{
297   \detokenize{#1}~could~not~find~file~#2
298 }
```

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
297 \cs_new_protected:Nn \stex_path_from_string:Nn {
298   \str_set:Nx \l_tmpa_str { #2 }
299   \str_if_empty:NTF \l_tmpa_str {
300     \seq_clear:N #1
301   }{
302     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
303     \sys_if_platform_windows:T{
304       \seq_clear:N \l_tmpa_tl
305       \seq_map_inline:Nn #1 {
306         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
307         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl
308       }
309     }
310   }
```

```

308     }
309     \seq_set_eq:NN #1 \l_tmpa_tl
310   }
311   \stex_path_canonicalize:N #1
312 }
313 }
314 \cs_generate_variant:Nn \stex_path_from_string:Nn
315 { 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
316 \cs_new_protected:Nn \stex_path_to_string:NN {
317   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
318 }
319
320 \cs_new:Nn \stex_path_to_string:N {
321   \seq_use:Nn #1 /
322 }

```

(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
323 \str_const:Nn \c__stex_path_dot_str {.}
324 \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.

```

325 \cs_new_protected:Nn \stex_path_canonicalize:N {
326   \seq_if_empty:NF #1 {
327     \seq_clear:N \l_tmpa_seq
328     \seq_get_left:NN #1 \l_tmpa_tl
329     \str_if_empty:NT \l_tmpa_tl {
330       \seq_put_right:Nn \l_tmpa_seq {}
331     }
332     \seq_map_inline:Nn #1 {
333       \str_set:Nn \l_tmpa_tl { ##1 }
334       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
335         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
336           \seq_if_empty:NTF \l_tmpa_seq {
337             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
338               \c__stex_path_up_str
339             }
340           }{
341             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
342             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
343               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
344                 \c__stex_path_up_str
345               }
346             }{
347               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
348             }

```

```

349     }
350   }{
351     \str_if_empty:NF \l_tmpa_tl {
352       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
353     }
354   }
355 }
356 }
357 \seq_gset_eq:NN #1 \l_tmpa_seq
358 }
359 }

```

(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`

```

360 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
361   \seq_if_empty:NTF #1 {
362     \prg_return_false:
363   }{
364     \seq_get_left:NN #1 \l_tmpa_tl
365     \str_if_empty:NTF \l_tmpa_tl {
366       \prg_return_true:
367     }{
368       \prg_return_false:
369     }
370   }
371 }

```

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

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

372 \str_new:N\l_stex_kpsewhich_return_str
373 \cs_new_protected:Nn \stex_kpsewhich:n {
374   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
375   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
376   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
377 }

```

(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`

```

378 \sys_if_platform_windows:TF{
379   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
380 }{
381   \stex_kpsewhich:n{-var-value~PWD}
382 }
383
384 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
385 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
386 \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

387 <@@=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

388 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

389 `\str_set:Nx \c_stex_mainfile_str {\c_stex_pwd_str/\jobname.tex}`

390 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

391 `\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`.

```

392 \seq_gclear_new:N\g_stex_currentfile_seq
393 \AddToHook{file/before}{
394   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
395   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
396     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
397   }{
398     \stex_path_from_string:Nn\g_stex_currentfile_seq{
399       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
400     }
401   }
402   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
403   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
404 }
405 \AddToHook{file/after}{
406   \seq_if_empty:NF\g__stex_files_stack{
407     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
408   }
409   \seq_if_empty:NTF\g__stex_files_stack{
410     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
411   }{
412     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
413     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
414   }
415 }
```

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

19.4 MathHub Repositories

```

416 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
417 \str_if_empty:NTF\mathhub{
418   \stex_kpsewhich:n{-var-value~MATHHUB}
419   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
420
421   \str_if_empty:NTF\c_stex_mathhub_str{
422     \msg_warning:nn{stex}{warning/nomathhub}
423   }{
424     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
425     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
426   }
427 }{
428   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
429   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
430     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
431       \c_stex_pwd_str/\mathhub
432     }
433   }
434   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
435   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
436 }

```

(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
437 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
438   \str_set:Nx \l_tmpa_str { #1 }
439   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
440     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
441     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
442     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
443     \__stex_mathhub_find_manifest:N \l_tmpa_seq
444     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
445       \msg_error:nnxx{stex}{error/norepository}{#1}{
446         \stex_path_to_string:N \c_stex_mathhub_str
447       }
448     } {
449       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
450     }
451   }
452 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
453 \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`:

```

454 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
455   \seq_set_eq:NN \l_tmpa_seq #1
456   \bool_set_true:N \l_tmpa_bool
457   \bool_while_do:Nn \l_tmpa_bool {
458     \seq_if_empty:NTF \l_tmpa_seq {
459       \bool_set_false:N \l_tmpa_bool
460     }{
461       \file_if_exist:nTF{
462         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
463       }{
464         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
465         \bool_set_false:N \l_tmpa_bool
466       }{
467         \file_if_exist:nTF{
468           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
469         }{
470           \seq_put_right:Nn \l_tmpa_seq{META-INF}
471           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
472           \bool_set_false:N \l_tmpa_bool
473         }{
474           \file_if_exist:nTF{
475             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
476           }{
477             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
478             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
479             \bool_set_false:N \l_tmpa_bool
480           }{
481             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
482           }
483         }
484       }
485     }
486   }
487   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
488 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

489 \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:

```

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

```

```

498 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
499 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
500 }
501 \exp_args:No \str_case:nnTF \l_tmpa_tl {
502 {id} {
503 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
504 { id } \l_tmpb_tl
505 }
506 {narration-base} {
507 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
508 { narr } \l_tmpb_tl
509 }
510 {url-base} {
511 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
512 { docurl } \l_tmpb_tl
513 }
514 {source-base} {
515 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
516 { ns } \l_tmpb_tl
517 }
518 {ns} {
519 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
520 { ns } \l_tmpb_tl
521 }
522 {dependencies} {
523 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
524 { deps } \l_tmpb_tl
525 }
526 }{}{}
527 }{}
528 }
529 \ior_close:N \c__stex_mathhub_manifest_ior
530 }

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

531 \cs_new_protected:Nn \stex_set_current_repository:n {
532 \stex_require_repository:n { #1 }
533 \prop_set_eq:Nc \l_stex_current_repository_prop {
534 c_stex_mathhub_#1_manifest_prop
535 }
536 }

```

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

`\stex_require_repository:n`

```

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

```



```

545     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
546     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
547   }
548 }
549 }
550 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

```

567 \cs_new_protected:Nn \stex_in_repository:nn {
568   \str_set:Nx \l_tmpa_str { #1 }
569   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
570   \str_if_empty:NTF \l_tmpa_str {
571     \exp_args:Ne \l_tmpa_cs{
572       \prop_item:Nn \l_stex_current_repository_prop { id }
573     }
574   }{
575     \stex_require_repository:n \l_tmpa_str
576     \str_set:Nx \l_tmpa_str { #1 }
577     \exp_args:Nne \use:nn {
578       \stex_set_current_repository:n \l_tmpa_str
579       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
580     }{
581       \stex_set_current_repository:n {
582         \prop_item:Nn \l_stex_current_repository_prop { id }
583       }
584     }
585   }
586 }

```

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

```

\inputref
\stex_inputref:nn
\mhinput\stex_mhinput:nn

587 \newif \ifinputref \inputreffalse
588
589 \cs_new_protected:Nn \stex_mhinput:nn {
590   \stex_in_repository:nn {#1} {
591     \ifinputref
592       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
593     \else
594       \inputreftrue
595       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
596     \inputreffalse
597   \fi
598 }
599 }
600 \NewDocumentCommand \mhinput { 0{} m}{
601   \stex_mhinput:nn{ #1 }{ #2 }
602 }
603
604 \cs_new_protected:Nn \stex_inputref:nn {
605   \stex_in_repository:nn {#1} {
606     \bool_lazy_any:nTF {
607       {\rustex_if_p:} {\latexml_if_p:}
608     } {
609       \str_clear:N \l_tmpa_str
610       \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
611         \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
612       }
613       \stex_annotate_invisible:nnn{inputref}{
614         \l_tmpa_str / #2
615       }{}
616     }{
617       \begingroup
618         \inputreftrue
619         \input{ \c_stex_mathhub_str / ##1 / source / #2 }
620       \endgroup
621     }
622   }
623 }
624
625 \NewDocumentCommand \inputref { 0{} m}{
626   \stex_inputref:nn{ #1 }{ #2 }
627 }
628
629 \cs_new_protected:Nn \stex_mhbibresource:nn {
630   \stex_in_repository:nn {#1} {
631     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
632   }
633 }
634 \newcommand\addmhbibresource[2] []{
635   \stex_mhbibresource:nn{ #1 }{ #2 }
636 }

```

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

\mhpath

```
637 \def \mhpath #1 #2 {
638   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
639     \c_stex_mathhub_str /
640     \prop_item:Nn \l_stex_current_repository_prop { id }
641     / source / #2
642   }{
643     \c_stex_mathhub_str / #1 / source / #2
644   }
645 }
```

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

\libinput

```
646 \cs_new_protected:Npn \libinput #1 {
647   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
648     \msg_error:nnn{stex}{error/notinarchive}\libinput
649   }
650   \bool_set_false:N \l_tmpa_bool
651   \tl_clear:N \l_tmpa_tl
652   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
653   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
654   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
655   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
656     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
657     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
658       / meta-inf / lib / #1.tex}{
659       \bool_set_true:N \l_tmpa_bool
660       \tl_put_right:Nx \l_tmpa_tl {
661         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
662           / meta-inf / lib / #1.tex}
663       }
664     }{}
665   }
666   \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
667     / \l_tmpa_str / lib / #1.tex
668   }{
669     \bool_set_true:N \l_tmpa_bool
670     \tl_put_right:Nx \l_tmpa_tl {
671       \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
672         / \l_tmpa_str / lib / #1.tex}
673     }
674   }{}
675   \bool_if:NF \l_tmpa_bool {
676     \msg_error:nnxx{stex}{error/nofile}{\exp_not:N\libinput}{#1.tex}
677   }
678   \l_tmpa_tl
679 }
```

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

```
680 \</package>
```

Chapter 20

STEX -References Implementation

```
681 <*package>
682
683 %%%%%%%%%% references.dtx %%%%%%%%%%
684
685 %\RequirePackage{hyperref}
686 %\RequirePackage{cleveref}
687 <@@=stex_refs>
688
689 Warnings and error messages
690
691 \iow_new:N \c__stex_refs_refs_iow
692 \AddToHook{begindocument}{
693   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
694 }
695 \AddToHook{enddocument}{
696   \iow_close:N \c__stex_refs_refs_iow
697 }
698
699 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
700
701 \NewDocumentCommand \STEXreftitle { m } {
702   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
703 }
704
```

20.1 Document URIs and URLs

```
702 \seq_new:N \g__stex_refs_all_refs_seq
703
704 \str_new:N \l_stex_current_docns_str
705
706 \cs_new_protected:Nn \stex_get_document_uri: {
707   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
708   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
709   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
710   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
711 }
```

```

711 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
712
713 \str_clear:N \l_tmpa_str
714 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
715   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
716 }
717
718 \str_if_empty:NTF \l_tmpa_str {
719   \str_set:Nx \l_stex_current_docns_str {
720     file:/\stex_path_to_string:N \l_tmpa_seq
721   }
722 }{
723   \bool_set_true:N \l_tmpa_bool
724   \bool_while_do:Nn \l_tmpa_bool {
725     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
726     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
727       {source} { \bool_set_false:N \l_tmpa_bool }
728     }{}{
729       \seq_if_empty:NT \l_tmpa_seq {
730         \bool_set_false:N \l_tmpa_bool
731       }
732     }
733   }
734
735   \seq_if_empty:NTF \l_tmpa_seq {
736     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
737   }{
738     \str_set:Nx \l_stex_current_docns_str {
739       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
740     }
741   }
742 }
743 }
744
745 \str_new:N \l_stex_current_docurl_str
746 \cs_new_protected:Nn \stex_get_document_url: {
747   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
748   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
749   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
750   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
751   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
752
753   \str_clear:N \l_tmpa_str
754   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
755     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
756       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
757     }
758   }
759
760   \str_if_empty:NTF \l_tmpa_str {
761     \str_set:Nx \l_stex_current_docurl_str {
762       file:/\stex_path_to_string:N \l_tmpa_seq
763     }
764   }{
765     \bool_set_true:N \l_tmpa_bool

```

```

765 \bool_while_do:Nn \l_tmpa_bool {
766   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
767   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
768     {source} { \bool_set_false:N \l_tmpa_bool }
769   }{}{
770     \seq_if_empty:NT \l_tmpa_seq {
771       \bool_set_false:N \l_tmpa_bool
772     }
773   }
774 }
775
776 \seq_if_empty:NTF \l_tmpa_seq {
777   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
778 }{
779   \str_set:Nx \l_stex_current_docurl_str {
780     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
781   }
782 }
783 }
784 }

```

20.2 Setting Reference Targets

```

785 \str_const:Nn \c__stex_refs_url_str{URL}
786 \str_const:Nn \c__stex_refs_ref_str{REF}
787 % @currentlabel -> number
788 % @currentlabelname -> title
789 % @currentHref -> name.number <- id of some kind
790 % \theH# -> \arabic{section}
791 % \the# -> number
792 % \hyper@makecurrent{#}
793 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
794   \stex_get_document_uri:
795   \str_set:Nx \l_tmpa_str { #1 }
796   \str_if_empty:NT \l_tmpa_str {
797     \int_zero:N \l_tmpa_int
798     \bool_set_true:N \l_tmpa_bool
799     \bool_while_do:Nn \l_tmpa_bool {
800       \cs_if_exist:cTF {
801         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
802       }{
803         \int_incr:N \l_tmpa_int
804       }{
805         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
806         \bool_set_false:N \l_tmpa_bool
807       }
808     }
809   }
810   \str_set:Nx \l_tmpa_str {
811     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
812   }
813   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
814   \stex_if_smsmode:TF {
815     \stex_get_document_url:

```

```

816 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
817 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
818 }{
819 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}{
820 \exp_args:Nx\label{sref_\l_tmpa_str}
821 \str_gset:cx {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
822 }
823 }

824 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
825 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
826 }

```

20.3 Using References

```

827 \str_new:N \l__stex_refs_indocument_str
828 \keys_define:nn { stex / sref } {
829 linktext .tl_set:N = \l__stex_refs_linktext_tl ,
830 fallback .tl_set:N = \l__stex_refs_fallback_tl ,
831 pre .tl_set:N = \l__stex_refs_pre_tl ,
832 post .tl_set:N = \l__stex_refs_post_tl ,
833 %indoc .str_set_x:N = \l__stex_refs_repo_str ,
834 }
835
836 \bool_new:N \c__stex_refs_hyperref_bool
837 \bool_set_false:N \c__stex_refs_hyperref_bool
838 \AddToHook{begindocument}{
839 \@ifpackageloaded{hyperref}{
840 \bool_set_true:N \c__stex_refs_hyperref_bool
841 }{}
842 }
843
844
845 \cs_new_protected:Nn \__stex_refs_args:n {
846 \tl_clear:N \l__stex_refs_linktext_tl
847 \tl_clear:N \l__stex_refs_fallback_tl
848 \tl_clear:N \l__stex_refs_pre_tl
849 \tl_clear:N \l__stex_refs_post_tl
850 \str_clear:N \l__stex_refs_repo_str
851 \keys_set:nn { stex / sref } { #1 }
852 }
853
854 \NewDocumentCommand \sref { 0{} m }{
855 \__stex_refs_args:n { #1 }
856 \str_if_empty:NTF \l__stex_refs_indocument_str {
857 \str_set:Nn \l_tmpa_str { #2 }
858 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
859 \tl_set:Nn \l_tmpa_tl {
860 \l__stex_refs_fallback_tl
861 }
862 \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
863 \str_set:Nn \l_tmpb_str { ##1 }
864 \str_if_eq:eeT { \l_tmpa_str } {
865 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{ -1 }
866 } {

```

```

867 \seq_map_break:n {
868   \tl_set:Nn \l_tmpa_tl {
869     % doc uri in \l_tmpb_str
870     \str_set:Nx \l_tmpa_str {\use:c{sref_\l_tmpb_str_type}}
871     \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
872       % reference
873       \cs_if_exist:cTF{autoref}{
874         \l__stex_refs_pre_tl\autoref{sref_\l_tmpb_str}\l__stex_refs_post_tl
875       }{
876         \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
877       }
878     }{
879       % URL
880       \if_bool:N \c__stex_refs_hyperref_bool {
881         \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
882       }{
883         \l__stex_refs_fallback_tl
884       }
885     }
886   }
887 }
888 }
889 }
890 \l_tmpa_tl
891 }{
892   % TODO
893 }
894 }
895
896 </package>

```


Chapter 21

STEX -Modules Implementation

```
897 <*package>
898
899 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
900
901 <@@=stex_modules>
902
903   Warnings and error messages
904   \msg_new:nnn{stex}{error/unknownmodule}{
905     No~module~#1~found
906   }
907   \msg_new:nnn{stex}{error/syntax}{
908     Syntax~error:~#1
909   }
910   \msg_new:nnn{stex}{error/siglanguage}{
911     Module~#1~declares~signature~#2,~but~does~not~
912     declare~its~language
913   }
```

\l_stex_current_module_str The current module:

```
912 \str_new:N \l_stex_current_module_str
```

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

\l_stex_all_modules_seq Stores all available modules

```
913 \seq_new:N \l_stex_all_modules_seq
```

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

\stex_if_in_module_p:

\stex_if_in_module:TF

```
914 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
915   \str_if_empty:NTF \l_stex_current_module_str
916   \prg_return_false: \prg_return_true:
917 }
```

(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`

```
918 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
919   \prop_if_exist:cnTF { c_stex_module_#1_prop }
920   \prg_return_true: \prg_return_false:
921 }
```

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

`\stex_add_to_current_module:n`
`\STEXexport`

Only allowed within modules:

```
922 \cs_new_protected:Nn \stex_add_to_current_module:n {
923   \prop_get:cnN {c_stex_module_\l_stex_current_module_str_prop} { content } \l_tmpa_tl
924   \tl_put_right:Nn \l_tmpa_tl { #1 }
925   \prop_gput:cno {c_stex_module_\l_stex_current_module_str_prop} { content } { \l_tmpa_tl }
926 }
927 \cs_new_protected:Npn \STEXexport {
928   \begingroup
929   \newlinechar=-1\relax
930   \endlinechar=-1\relax
931   %\catcode'\ = 9\relax
932   \expandafter\endgroup\STEXexport:n
933 }
934 \cs_new_protected:Nn \STEXexport:n {
935   \ignorespaces #1
936   \stex_add_to_current_module:n { \ignorespaces #1 }
937   \stex_smsmode_set_codes:
938 }
939 \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`

```
940 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
941   \str_set:Nx \l_tmpa_str { #1 }
942   \prop_get:cnN {c_stex_module_\l_stex_current_module_str_prop} { constants } \l_tmpa_seq
943   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
944   \prop_gput:cno {c_stex_module_\l_stex_current_module_str_prop} { constants } \l_tmpa_seq
945 }
```

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

`\stex_add_import_to_current_module:n`

```
946 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
947   \str_set:Nx \l_tmpa_str { #1 }
948   \prop_get:cnN {c_stex_module_\l_stex_current_module_str_prop} { imports } \l_tmpa_seq
949   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
950   \prop_gput:cno {c_stex_module_\l_stex_current_module_str_prop} { imports } \l_tmpa_seq
951 }
```

(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).

```

952 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
953   \str_set:Nx \l_tmpa_str { #1 }
954   \seq_set_eq:NN \l_tmpa_seq #2
955   % split off file extension
956   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
957   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
958   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
959   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
960
961   \bool_set_true:N \l_tmpa_bool
962   \bool_while_do:Nn \l_tmpa_bool {
963     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
964     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
965       {source} { \bool_set_false:N \l_tmpa_bool }
966     }{}{
967       \seq_if_empty:NT \l_tmpa_seq {
968         \bool_set_false:N \l_tmpa_bool
969       }
970     }
971   }
972
973   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
974   \str_if_empty:NTF \l_stex_modules_subpath_str {
975     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
976   }{
977     \str_set:Nx \l_stex_modules_ns_str {
978       \l_tmpa_str/\l_stex_modules_subpath_str
979     }
980   }
981 }

```

(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
\l_stex_modules_subpath_str
982 \str_new:N \l_stex_modules_ns_str
983 \str_new:N \l_stex_modules_subpath_str

```

(End definition for `\l_stex_modules_ns_str` and `\l_stex_modules_subpath_str`. These variables are documented on page ??.)

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

```

984 \cs_new_protected:Nn \stex_modules_current_namespace: {
985   \str_clear:N \l_stex_modules_subpath_str
986   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
987     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
988   }{
989     % split off file extension
990     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
991     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
992     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str

```

```

993     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
994     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
995     \str_set:Nx \l_stex_modules_ns_str {
996         file:/\stex_path_to_string:N \l_tmpa_seq
997     }
998 }
999 }

```

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

21.1 The module environment

module arguments:

```

1000 \keys_define:nn { stex / module } {
1001     title      .str_set_x:N = \l_stex_module_title_str ,
1002     ns         .str_set_x:N = \l_stex_module_ns_str ,
1003     lang       .str_set_x:N = \l_stex_module_lang_str ,
1004     sig        .str_set_x:N = \l_stex_module_sig_str ,
1005     creators   .str_set_x:N = \l_stex_module_creators_str ,
1006     contributors .str_set_x:N = \l_stex_module_contributors_str ,
1007     meta       .str_set_x:N = \l_stex_module_meta_str ,
1008     srccite    .str_set_x:N = \l_stex_module_srccite_str
1009 }
1010
1011 \cs_new_protected:Nn \__stex_modules_args:n {
1012     \str_clear:N \l_stex_module_title_str
1013     \str_clear:N \l_stex_module_ns_str
1014     \str_clear:N \l_stex_module_lang_str
1015     \str_clear:N \l_stex_module_sig_str
1016     \str_clear:N \l_stex_module_creators_str
1017     \str_clear:N \l_stex_module_contributors_str
1018     \str_clear:N \l_stex_module_meta_str
1019     \str_clear:N \l_stex_module_srccite_str
1020     \keys_set:nn { stex / module } { #1 }
1021 }
1022
1023 % module parameters here? In the body?
1024

```

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

```

1025 \cs_new_protected:Nn \stex_module_setup:nn {
1026     \str_set:Nx \l_stex_module_name_str { #2 }
1027     \__stex_modules_args:n { #1 }

```

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

Are we in a nested module?

```

1028 \stex_if_in_module:TF {
1029     % Nested module
1030     \prop_get:cnN {c_stex_module\_l_stex_current_module_str _prop}
1031         { ns } \l_stex_module_ns_str
1032     \str_set:Nx \l_stex_module_name_str {
1033         \prop_item:cn {c_stex_module\_l_stex_current_module_str _prop}
1034         { name } / \l_stex_module_name_str

```

```

1035   }
1036 }{
1037   % not nested:
1038   \str_if_empty:NT \l_stex_module_ns_str {
1039     \stex_modules_current_namespace:
1040     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1041     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1042       / {\l_stex_module_ns_str}
1043     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1044     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1045       \str_set:Nx \l_stex_module_ns_str {
1046         \stex_path_to_string:N \l_tmpa_seq
1047       }
1048     }
1049   }
1050 }

```

Next, we determine the language of the module:

```

1051 \str_if_empty:NT \l_stex_module_lang_str {
1052   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1053   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1054   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1055   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1056   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1057     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1058       inferred~from~file~name}
1059     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1060   }
1061 }
1062
1063 \str_if_empty:NF \l_stex_module_lang_str {
1064   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1065   \l_tmpa_str {
1066     \ltx@ifpackageloaded{babel}{
1067       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1068     }{}
1069   } {
1070     \msg_error:nxn{stex}{error/unknownlanguage}{\l_tmpa_str}
1071   }
1072 }

```

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

```

1073 \str_if_empty:NTF \l_stex_module_sig_str {
1074   \str_clear:N \l_tmpa_str
1075   \seq_clear:N \l_tmpa_seq
1076   \tl_clear:N \l_tmpa_tl
1077   \exp_args:Nnx \prop_gset_from_keyval:cn {
1078     c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1079   } {
1080     name      = \l_stex_module_name_str ,
1081     ns        = \l_stex_module_ns_str ,
1082     imports   = \exp_not:o { \l_tmpa_seq } ,
1083     constants = \exp_not:o { \l_tmpa_seq } ,

```

```

1084     content    = \exp_not:o { \l_tmpa_tl } ,
1085     file       = \exp_not:o { \g_stex_currentfile_seq } ,
1086     lang       = \l_stex_module_lang_str ,
1087     sig        = \l_stex_module_sig_str ,
1088     meta       = \l_stex_module_meta_str
1089   }
1090   \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1091   \str_if_empty:NT \l_stex_module_meta_str {
1092     \str_set:Nx \l_stex_module_meta_str {
1093       \c_stex_metatheory_ns_str ? Metatheory
1094     }
1095   }
1096   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1097     \exp_args:Nx \stex_add_to_current_module:n {
1098       \stex_activate_module:n {\l_stex_module_meta_str}
1099     }
1100     \stex_activate_module:n {\l_stex_module_meta_str}
1101   }
1102 }{
1103   \str_if_empty:NT \l_stex_module_lang_str {
1104     \msg_error:nnxx{stex}{error/siglanguage}{
1105       \l_stex_module_ns_str?\l_stex_module_name_str
1106     }{\l_stex_module_sig_str}
1107   }
1108
1109   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1110   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1111   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1112   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1113   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1114   \str_set:Nx \l_tmpa_str {
1115     \stex_path_to_string:N \l_tmpa_seq /
1116     \l_tmpa_str . \l_stex_module_sig_str .tex
1117   }
1118   \IfFileExists \l_tmpa_str {
1119     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1120       \seq_clear:N \l_stex_all_modules_seq
1121       %\prop_clear:N \l_stex_current_module_prop
1122       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1123       \input { \l_tmpa_str }
1124     }
1125   }{
1126     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1127   }
1128   \stex_activate_module:n {
1129     \l_stex_module_ns_str ? \l_stex_module_name_str
1130   }
1131   %\prop_set_eq:Nc \l_stex_current_module_prop {
1132   %   c_stex_module_
1133   %   \l_stex_module_ns_str ?
1134   %   \l_stex_module_name_str
1135   %   _prop

```

```

1136     %}
1137     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1138   }
1139 }

```

(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}

1140 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1141   \stex_reactivate_macro:N \STEXexport
1142   \stex_reactivate_macro:N \importmodule
1143   \stex_reactivate_macro:N \symdecl
1144   \stex_reactivate_macro:N \notation
1145   \stex_reactivate_macro:N \symdef
1146   \stex_module_setup:nn{#1}{#2}
1147
1148   \stex_debug:nn{modules}{
1149     New~module:~\
1150     Namespace:~\l_stex_module_ns_str~\
1151     Name:~\l_stex_module_name_str~\
1152     Language:~\l_stex_module_lang_str~\
1153     Signature:~\l_stex_module_sig_str~\
1154     Metatheory:~\l_stex_module_meta_str~\
1155     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1156   }
1157
1158   \seq_put_right:Nx \l_stex_all_modules_seq {
1159     \l_stex_module_ns_str ? \l_stex_module_name_str
1160   }
1161
1162   % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1163   %   { \l_stex_module_ns_str ? \l_stex_module_name_str }
1164
1165   \stex_if_smsmode:TF {
1166     \stex_smsmode_set_codes:
1167   } {
1168     \begin{stex_annotate_env} {theory} {
1169       \l_stex_module_ns_str ? \l_stex_module_name_str
1170     }
1171
1172     \stex_annotate_invisible:nnn{header}{} {
1173       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1174       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1175       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1176         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1177       }
1178     }
1179   }
1180   % TODO: Inherit metatheory for nested modules?
1181 }
1182 \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}

1183 \cs_new_protected:Nn \__stex_modules_end_module: {
1184 % \str_set:Nx \l_tmpa_str {
1185 %   c_stex_module_
1186 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1187 %   \prop_item:Nn \l_stex_current_module_prop { name }
1188 %   _prop
1189 % }
1190 %^^A \prop_new:c { \l_tmpa_str }
1191 % \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1192 \stex_debug:nn{modules}{Closing~module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1193 }

```

(End definition for __stex_modules_end_module:.)

@module The core environment, with no header

```

1194 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1195 \NewDocumentEnvironment { @module } { 0{} m } {
1196   \par
1197   \__stex_modules_begin_module:nn{#1}{#2}
1198 } {
1199   \__stex_modules_end_module:
1200   \stex_if_smsmode:TF {
1201 %     \exp_args:Nx \stex_add_to_sms:n {
1202 %       \prop_gset_from_keyval:cn {
1203 %         c_stex_module_
1204 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1205 %         \prop_item:Nn \l_stex_current_module_prop { name }
1206 %         _prop
1207 %       } {
1208 %         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1209 %         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1210 %         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1211 %         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1212 %         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1213 %         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1214 %         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1215 %         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1216 %         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1217 %       }
1218 %     }
1219   }{
1220     \end{stex_annotate_env}
1221   }
1222 }

```

\stex_modules_heading: Code for document headers

```

1223 \cs_if_exist:NTF \thesection {
1224   \newcounter{module}[section]
1225 }{
1226   \newcounter{module}
1227 }
1228

```



```

1229 \bool_if:NT \c_stex_showmods_bool {
1230   \latexml_if:F { \RequirePackage{mdframed} }
1231 }
1232
1233 \cs_new_protected:Nn \stex_modules_heading: {
1234   \stepcounter{module}
1235   \par
1236   \bool_if:NT \c_stex_showmods_bool {
1237     \noindent{\textbf{Module} ~
1238       \cs_if_exist:NT \thesection {\thesection.}
1239       \themodule ~ [\l_stex_module_name_str]
1240     }
1241     \str_if_empty:NTF \l_stex_module_title_str {
1242       }{
1243         \quad(\l_stex_module_title_str)\hfill
1244       }\par
1245     }
1246     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1247     % TODO
1248     \stex_ref_new_doc_target:n \l_stex_module_name_str
1249   }

```

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

Finally:

```

1250 \NewDocumentEnvironment { module } { 0 } { m } {
1251   \bool_if:NT \c_stex_showmods_bool {
1252     \begin{mdframed}
1253   }
1254   \begin{@module}[\#1]{\#2}
1255   \stex_modules_heading:
1256 }{
1257   \end{@module}
1258   \bool_if:NT \c_stex_showmods_bool {
1259     \end{mdframed}
1260   }
1261 }

```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1262 \NewDocumentCommand \STEXModule { m } {
1263   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1264   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1265   \tl_set:Nn \l_tmpa_tl {
1266     \msg_error:nnx{stex}{error/unknownmodule}{\#1}
1267   }
1268   \seq_map_inline:Nn \l_stex_all_modules_seq {
1269     \str_set:Nn \l_tmpb_str { ##1 }
1270     \str_if_eq:eeT { \l_tmpa_str } {
1271       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1272     } {
1273       \seq_map_break:n {
1274         \tl_set:Nn \l_tmpa_tl {

```

```

1275         \stex_invoke_module:n { ##1 }
1276     }
1277 }
1278 }
1279 }
1280 \l_tmpa_tl
1281 }
1282
1283 \cs_new_protected:Nn \stex_invoke_module:n {
1284     \stex_debug:nn{modules}{Invoking~module~#1}
1285     \peek_charcode_remove:NTF ! {
1286         \__stex_modules_invoke_uri:nN { #1 }
1287     } {
1288         \peek_charcode_remove:NTF ? {
1289             \__stex_modules_invoke_symbol:nn { #1 }
1290         } {
1291             \msg_error:nnx{stex}{error/syntax}{
1292                 ?~or~!~expected~after~
1293                 \c_backslash_str STEXModule{#1}
1294             }
1295         }
1296     }
1297 }
1298
1299 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1300     \str_set:Nn #2 { #1 }
1301 }
1302
1303 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1304     \stex_invoke_symbol:n{#1?#2}
1305 }

```

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

`\stex_activate_module:n`

```

1306 \cs_new_protected:Nn \stex_activate_module:n {
1307     \stex_debug:nn{modules}{Activating~module~#1}
1308     \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1309         \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1310         \prop_item:cn { c_stex_module_#1_prop } { content }
1311     }
1312 }

```

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

```

1313 </package>

```

Chapter 22

STEX -Module Inheritance Implementation

```
1314 ⟨*package⟩
1315
1316 %%%%%%%%% inheritance.dtx %%%%%%%%%
1317
```

22.1 SMS Mode

```
1318 ⟨@@=stex_smsmode⟩

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1319 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1320 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1321 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1322
1323 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1324   \makeatletter
1325   \makeatother
1326   \ExplSyntaxOn
1327   \ExplSyntaxOff
1328 }
1329
1330 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1331   \symdef
1332   \importmodule
1333   \notation
1334   \symdecl
1335   \STEXexport
1336 }
1337
1338 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1339   \tl_to_str:n {
1340     module,
1341     @module
```

```

1342 }
1343 }

```

(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

```

```

1344 \bool_new:N \g__stex_smsmode_bool
1345 \bool_set_false:N \g__stex_smsmode_bool
1346 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1347   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1348 }

```

(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

```

```

1349 \bool_new:N \g__stex_smsmode_catcode_bool
1350 \bool_set_false:N \g__stex_smsmode_catcode_bool
1351 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1352   \bool_if:NTF \g__stex_smsmode_catcode_bool
1353   \prg_return_true: \prg_return_false:
1354 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1355 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1356   \stex_if_smsmode:T {
1357     \__stex_smsmode_if_catcodes:F {
1358       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1359       \exp_after:wN \char_gset_active_eq:NN
1360       \c_backslash_str \__stex_smsmode_cs:
1361       \tex_global:D \char_set_catcode_active:N \
1362       \tex_global:D \char_set_catcode_other:N $
1363       \tex_global:D \char_set_catcode_other:N ^
1364       \tex_global:D \char_set_catcode_other:N _
1365       \tex_global:D \char_set_catcode_other:N &
1366       \tex_global:D \char_set_catcode_other:N ##
1367     }
1368   }
1369 } \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.

```

1370 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1371   \__stex_smsmode_if_catcodes:T {
1372     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1373     \exp_after:wN \tex_global:D \exp_after:wN
1374     \char_set_catcode_escape:N \c_backslash_str
1375     \tex_global:D \char_set_catcode_math_toggle:N $
1376     \tex_global:D \char_set_catcode_math_superscript:N ^
1377     \tex_global:D \char_set_catcode_math_subscript:N _
1378     \tex_global:D \char_set_catcode_alignment:N &
1379     \tex_global:D \char_set_catcode_parameter:N ##
1380   }
1381 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1382 \cs_new_protected:Nn \stex_in_smsmode:nn {
1383   \vbox_set:Nn \l_tmpa_box {
1384     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1385     \bool_gset_true:N \g__stex_smsmode_bool
1386     \stex_smsmode_set_codes:
1387     #2
1388     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1389     \stex_if_smsmode:F {
1390       \__stex_smsmode_unset_codes:
1391     }
1392   }
1393   \box_clear:N \l_tmpa_box
1394 }

```

(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:

```

1395 \cs_new_protected:Nn \_stex_smsmode_cs: {
1396   \str_clear:N \l_tmpa_str
1397   \peek_analysis_map_inline:n {
1398     % #1: token (one expansion)
1399     % #2: charcode
1400     % #3 catcode
1401     \token_if_eq_charcode:NNTF ##3 B {
1402       % token is a letter
1403       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1404     } {
1405       \str_if_empty:NTF \l_tmpa_str {
1406         % we don't allow (or need) single non-letter CSs
1407         % for now
1408         \peek_analysis_map_break:
1409       }{
1410         \str_if_eq:onTF \l_tmpa_str { begin } {
1411           \peek_analysis_map_break:n {
1412             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1413           }
1414         } {
1415           \str_if_eq:onTF \l_tmpa_str { end } {
1416             \peek_analysis_map_break:n {
1417               \exp_after:wN \_stex_smsmode_checkend:n ##1
1418             }
1419           } {
1420             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1421             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1422             \g_stex_smsmode_allowedmacros_tl
1423             { \use:c{\l_tmpa_str} } {
1424               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1425               \peek_analysis_map_break:n {
1426                 \exp_after:wN \l_tmpa_tl ##1
1427               }

```

```

1428     } {
1429         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1430         \g_stex_smsmode_allowedmacros_escape_tl
1431         { \use:c{\l_tmpa_str} } {
1432             \__stex_smsmode_unset_codes:
1433             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1434             % TODO \__stex_smsmode_rescan_cs:
1435             % \int_compare:nNnTF {##2} = {92} {
1436             %     \peek_analysis_map_break:n {
1437             %         \__stex_smsmode_unset_codes:
1438             %         \__stex_smsmode_rescan_cs:
1439             %     }
1440             % } {
1441             %     \peek_analysis_map_break:n {
1442             %         \exp_after:wN \l_tmpa_tl ##1
1443             %     }
1444             % }
1445             } {
1446                 \int_compare:nNnTF {##2} = {92} {
1447                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1448                 }{
1449                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1450                 }
1451             }
1452         }
1453     }
1454 }
1455 }
1456 }
1457 }
1458 }

```

(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:

```

1459 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1460     \str_clear:N \l_tmpb_str
1461     \peek_analysis_map_inline:n {
1462         \token_if_eq_charcode:NNTF ##3 B {
1463             % token is a letter
1464             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1465         } {
1466             \peek_analysis_map_break:n {
1467                 \exp_after:wN \use:c \exp_after:wN {
1468                     \exp_after:wN \l_tmpa_str\exp_after:wN
1469                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1470             }
1471         }
1472     }
1473 }

```

(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.

```

1474 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1475   \str_set:Nn \l_tmpa_str { #1 }
1476   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1477     \__stex_smsmode_unset_codes:
1478     \begin{#1}
1479   }
1480 }
```

(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.

```

1481 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1482   \str_set:Nn \l_tmpa_str { #1 }
1483   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1484     \end{#1}
1485   }
1486 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

1487 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1488 \cs_new_protected:Nn \stex_import_module_uri:nn {
1489   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1490   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1491
1492   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1493   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1494   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1495
1496   \stex_modules_current_namespace:
1497   \bool_lazy_all:nTF {
1498     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1499     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1500     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1501   }{
1502     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1503     \str_set_eq:NN \l_stex_module_ns
1504   }{
1505     \str_if_empty:NT \l__stex_importmodule_archive_str {
1506       \prop_if_empty:NF \l_stex_current_repository_prop {
1507         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1508       }
1509     }
1510     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1511       \str_if_empty:NF \l__stex_importmodule_path_str {
1512         \str_set:Nx \l_stex_module_ns_str {
1513           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1514         }
1515       }
1516     }
```

```

1516   }{
1517     \stex_require_repository:n \l__stex_importmodule_archive_str
1518     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1519       \l_stex_module_ns_str
1520     \str_if_empty:NF \l__stex_importmodule_path_str {
1521       \str_set:Nx \l_stex_module_ns_str {
1522         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1523       }
1524     }
1525   }
1526 }
1527 }

```

(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>	1528 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1529 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1530 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1531 <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>}
1532 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1533   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1534
1535     % archive
1536     \str_set:Nx \l_tmpa_str { #2 }
1537     \str_if_empty:NTF \l_tmpa_str {
1538       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1539     } {
1540       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1541       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1542       \seq_put_right:Nn \l_tmpa_seq { source }
1543     }
1544
1545     % path
1546     \str_set:Nx \l_tmpb_str { #3 }
1547     \str_if_empty:NTF \l_tmpb_str {
1548       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1549
1550       \ltx@ifpackageloaded{babel} {
1551         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1552           { \language } \l_tmpb_str {
1553           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1554         }
1555       } {
1556         \str_clear:N \l_tmpb_str
1557       }
1558
1559       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1560       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1561         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```



```

1562     }{
1563       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1564       \IfFileExists{ \l_tmpa_str.tex }{
1565         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1566       }{
1567         % try english as default
1568         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1569         \IfFileExists{ \l_tmpa_str.en.tex }{
1570           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1571         }{
1572           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1573         }
1574       }
1575     }
1576
1577   } {
1578     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1579     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1580
1581     \ltx@ifpackageloaded{babel} {
1582       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1583       { \language } \l_tmpb_str {
1584         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1585       }
1586     } {
1587       \str_clear:N \l_tmpb_str
1588     }
1589
1590     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1591
1592     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1593     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1594       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1595     }{
1596       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1597       \IfFileExists{ \l_tmpa_str/#4.tex }{
1598         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1599       }{
1600         % try english as default
1601         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1602         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1603           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1604         }{
1605           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1606           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1607             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1608           }{
1609             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1610             \IfFileExists{ \l_tmpa_str.tex }{
1611               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1612             }{
1613               % try english as default
1614               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1615               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1616         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1617     }{
1618         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1619     }
1620 }
1621 }
1622 }
1623 }
1624 }
1625 }
1626
1627 % \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1628 % \seq_clear:N \g_stex_modules_in_file_seq
1629 % \exp_args:Nnx \use:nn {
1630     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1631         \seq_clear:N \l_stex_all_modules_seq
1632         \str_clear:N \l_stex_current_module_str
1633         \str_set:Nx \l_tmpb_str { #2 }
1634         \str_if_empty:NF \l_tmpb_str {
1635             \stex_set_current_repository:n { #2 }
1636         }
1637         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1638         \input { \g__stex_importmodule_file_str }
1639     }
1640 % }{
1641
1642 % }
1643 % \prop_gput:Noo \g_stex_module_files_prop
1644 % \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1645 % \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1646
1647 \stex_if_module_exists:nF { #1 ? #4 } {
1648     \msg_error:nnx{stex}{error/unknownmodule}{
1649         #1?#4~(in~file~\g__stex_importmodule_file_str)
1650     }
1651 }
1652 }
1653 \stex_activate_module:n { #1 ? #4 }
1654 }

```

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

`\importmodule`

```

1655 \NewDocumentCommand \importmodule { 0{} m } {
1656     \stex_import_module_uri:nn { #1 } { #2 }
1657     \stex_debug:nn{modules}{Importing~module:~
1658         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1659     }
1660     \stex_if_smsmode:F {
1661         \stex_import_require_module:nnnn
1662         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1663         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1664         \stex_annotate_invisible:nnn
1665         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1666 }
1667 \exp_args:Nx \stex_add_to_current_module:n {
1668   \stex_import_require_module:nnnn
1669   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1670   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1671 }
1672 \exp_args:Nx \stex_add_import_to_current_module:n {
1673   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1674 }
1675 \stex_smsmode_set_codes:
1676 }
1677 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1678 \NewDocumentCommand \usemodule { 0{} m } {
1679   \stex_if_smsmode:F {
1680     \stex_import_module_uri:nn { #1 } { #2 }
1681     \stex_import_require_module:nnnn
1682     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1683     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1684     \stex_annotate_invisible:nnn
1685     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1686   }
1687   \stex_smsmode_set_codes:
1688 }

```

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

```

1689 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1690 <*package>
1691
1692 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1693
Warnings and error messages
1694
```

23.1 Symbol Declarations

```
1695 <@@=stex_symdecl>
\l_stex_all_symbols_seq Stores all available symbols
1696 \seq_new:N \l_stex_all_symbols_seq
(End definition for \l_stex_all_symbols_seq. This variable is documented on page 25.)

\STEXsymbol
1697 \NewDocumentCommand \STEXsymbol { m } {
1698   \stex_get_symbol:n { #1 }
1699   \exp_args:No
1700   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1701 }
(End definition for \STEXsymbol. This function is documented on page 27.)
symdecl arguments:
1702 \keys_define:nn { stex / symdecl } {
1703   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1704   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1705   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1706   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1707   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1708   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1709   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1710   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1711 }
```

```

1712
1713 \bool_new:N \l_stex_symdecl_make_macro_bool
1714
1715 \cs_new_protected:Nn \__stex_symdecl_args:n {
1716   \str_clear:N \l_stex_symdecl_name_str
1717   \str_clear:N \l_stex_symdecl_args_str
1718   \bool_set_false:N \l_stex_symdecl_local_bool
1719   \tl_clear:N \l_stex_symdecl_type_tl
1720   \tl_clear:N \l_stex_symdecl_definiens_tl
1721
1722   \keys_set:nn { stex / symdecl } { #1 }
1723 }

```

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

```

1724
1725 \NewDocumentCommand \symdecl { s O{} m } {
1726   \__stex_symdecl_args:n { #2 }
1727   \IfBooleanTF #1 {
1728     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1729   } {
1730     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1731   }
1732   \stex_symdecl_do:n { #3 }
1733   \stex_smsmode_set_codes:
1734 }
1735 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1736 \cs_new_protected:Nn \stex_symdecl_do:n {
1737   \stex_if_in_module:F {
1738     % TODO throw error? some default namespace?
1739   }
1740
1741   \str_if_empty:NT \l_stex_symdecl_name_str {
1742     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1743   }
1744
1745   \prop_if_exist:cT { g_stex_symdecl_
1746     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} {ns} ?
1747     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} {name} ?
1748     \l_stex_symdecl_name_str
1749     _prop
1750   }{
1751     % TODO throw error (beware of circular dependencies)
1752   }
1753
1754   \prop_clear:N \l_tmpa_prop
1755   \prop_put:Nnx \l_tmpa_prop { module } {
1756     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} {ns} ?
1757     \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop} {name}
1758   }

```

```

1759 \seq_clear:N \l_tmpa_seq
1760 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1761 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1762 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1763 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1764
1765 \exp_args:No \stex_add_constant_to_current_module:n {
1766   \l_stex_symdecl_name_str
1767 }
1768
1769 % arity/args
1770 \int_zero:N \l_tmpb_int
1771
1772 \bool_set_true:N \l_tmpa_bool
1773 \str_map_inline:Nn \l_stex_symdecl_args_str {
1774   \token_case_meaning:NnF ##1 {
1775     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1776     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1777     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1778     {\tl_to_str:n a} {
1779       \bool_set_false:N \l_tmpa_bool
1780       \int_incr:N \l_tmpb_int
1781     }
1782     {\tl_to_str:n B} {
1783       \bool_set_false:N \l_tmpa_bool
1784       \int_incr:N \l_tmpb_int
1785     }
1786   }{
1787     \msg_set:nnn{stex}{error/wrongargs}{
1788       args~value~in~symbol~declaration~for~
1789       \prop_item:Cn {c_stex_module\_l_stex_current_module_str _prop} {ns} ?
1790       \prop_item:Cn {c_stex_module\_l_stex_current_module_str _prop} {name} ?
1791       \l_stex_symdecl_name_str ~
1792       needs~to~be~
1793       i,~a,~b~or~B,~but~##1~given
1794     }
1795     \msg_error:nn{stex}{error/wrongargs}
1796   }
1797 }
1798 \bool_if:NTF \l_tmpa_bool {
1799   % possibly numeric
1800   \str_if_empty:NTF \l_stex_symdecl_args_str {
1801     \prop_put:Nnn \l_tmpa_prop { args } {}
1802     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1803   }{
1804     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1805     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1806     \str_clear:N \l_tmpa_str
1807     \int_step_inline:nn \l_tmpa_int {
1808       \str_put_right:Nn \l_tmpa_str i
1809     }
1810     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1811   }
1812 } {

```

```

1813 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1814 \prop_put:Nnx \l_tmpa_prop { arity }
1815 { \str_count:N \l_stex_symdecl_args_str }
1816 }
1817 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1818
1819
1820 % semantic macro
1821
1822 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1823   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1824     \prop_item:Nn \l_tmpa_prop { module } ?
1825     \prop_item:Nn \l_tmpa_prop { name }
1826   } }
1827
1828   \bool_if:NF \l_stex_symdecl_local_bool {
1829     \exp_args:Nx \stex_add_to_current_module:n {
1830       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1831         \prop_item:Nn \l_tmpa_prop { module } ?
1832         \prop_item:Nn \l_tmpa_prop { name }
1833       } }
1834     }
1835   }
1836 }
1837
1838 % add to all symbols
1839
1840 \bool_if:NF \l_stex_symdecl_local_bool {
1841   \exp_args:Nx \stex_add_to_current_module:n {
1842     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1843       \prop_item:Nn \l_tmpa_prop { module } ?
1844       \prop_item:Nn \l_tmpa_prop { name }
1845     }
1846   }
1847 }
1848
1849 \stex_debug:nn{symbols}{New~symbol:~
1850   \prop_item:Nn \l_tmpa_prop { module } ?
1851   \prop_item:Nn \l_tmpa_prop { name } ^^J
1852   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1853   Args:~\prop_item:Nn \l_tmpa_prop { args }
1854 }
1855
1856 % circular dependencies require this:
1857
1858 \prop_if_exist:cF {
1859   g_stex_symdecl_
1860   \prop_item:Nn \l_tmpa_prop { module } ?
1861   \prop_item:Nn \l_tmpa_prop { name }
1862   _prop
1863 } {
1864   \prop_gset_eq:cN {
1865     g_stex_symdecl_
1866     \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1867     \prop_item:Nn \l_tmpa_prop { name }
1868     _prop
1869   } \l_tmpa_prop
1870 }
1871
1872 \stex_if_smsmode:TF {
1873   \bool_if:NF \l_stex_symdecl_local_bool {
1874     \exp_args:Nx \stex_add_to_sms:n {
1875       \prop_gset_from_keyval:cn {
1876         g_stex_symdecl_
1877         \prop_item:Nn \l_tmpa_prop { module } ?
1878         \prop_item:Nn \l_tmpa_prop { name }
1879         _prop
1880       } {
1881         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1882         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1883         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1884         local     = \prop_item:Nn \l_tmpa_prop { local }      ,
1885         type      = \prop_item:Nn \l_tmpa_prop { type }       ,
1886         args      = \prop_item:Nn \l_tmpa_prop { args }       ,
1887         arity     = \prop_item:Nn \l_tmpa_prop { arity }      ,
1888         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1889       }
1890       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1891         \prop_item:Nn \l_tmpa_prop { module } ?
1892         \prop_item:Nn \l_tmpa_prop { name }
1893       }
1894     }
1895   }
1896 }{
1897   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1898     \prop_item:Nn \l_tmpa_prop { module } ?
1899     \prop_item:Nn \l_tmpa_prop { name }
1900   }
1901   \stex_if_do_html:T {
1902     \stex_annotate_invisible:nnn {symdecl} {
1903       \prop_item:Nn \l_tmpa_prop { module } ?
1904       \prop_item:Nn \l_tmpa_prop { name }
1905     } {
1906       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
1907       \stex_annotate_invisible:nnn{args}{}}{
1908         \prop_item:Nn \l_tmpa_prop { args }
1909       }
1910       \stex_annotate_invisible:nnn{macroname}{}{#1}
1911       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1912         \stex_annotate_invisible:nnn{definiens}{}
1913         {\l_stex_symdecl_definiens_tl$}
1914       }
1915     }
1916   }
1917 }
1918 }

```

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

`\stex_get_symbol:n`

```
1919 \str_new:N \l_stex_get_symbol_uri_str
1920
1921 \cs_new_protected:Nn \stex_get_symbol:n {
1922   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1923     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1924   }{
1925     % argument is a string
1926     % is it a command name?
1927     \cs_if_exist:cTF { #1 }{
1928       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1929       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1930       \str_if_empty:NTF \l_tmpa_str {
1931         \exp_args:Nx \cs_if_eq:NNTF {
1932           \tl_head:N \l_tmpa_tl
1933         } \stex_invoke_symbol:n {
1934           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1935         }{
1936           \__stex_symdecl_get_symbol_from_string:n { #1 }
1937         }
1938       } {
1939         \__stex_symdecl_get_symbol_from_string:n { #1 }
1940       }
1941     }{
1942       % argument is not a command name
1943       \__stex_symdecl_get_symbol_from_string:n { #1 }
1944       % \l_stex_all_symbols_seq
1945     }
1946   }
1947 }
1948
1949 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1950   \str_set:Nn \l_tmpa_str { #1 }
1951   \bool_set_false:N \l_tmpa_bool
1952   \stex_if_in_module:T {
1953     \prop_get:cnN {c_stex_module\l_stex_current_module_str _prop}
1954     { constants } \l_tmpa_seq
1955     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1956       \bool_set_true:N \l_tmpa_bool
1957       \str_set:Nx \l_stex_get_symbol_uri_str {
1958         \prop_item:cn {c_stex_module\l_stex_current_module_str _prop} { ns } ?
1959         \prop_item:cn {c_stex_module\l_stex_current_module_str _prop} { name } ? #1
1960       }
1961     }
1962   }
1963   \bool_if:NF \l_tmpa_bool {
1964     \tl_set:Nn \l_tmpa_tl {
1965       \msg_set:nnn{stex}{error/unknownsymbol}{
1966         No~symbol~#1~found!
1967       }
1968     }
1969     \msg_error:nn{stex}{error/unknownsymbol}
1970   }
1971   \str_set:Nn \l_tmpa_str { #1 }
1972   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1972 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1973   \str_set:Nn \l_tmpb_str { ##1 }
1974   \str_if_eq:eeT { \l_tmpa_str } {
1975     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1976   } {
1977     \seq_map_break:n {
1978       \tl_set:Nn \l_tmpa_tl {
1979         \str_set:Nn \l_stex_get_symbol_uri_str {
1980           ##1
1981         }
1982       }
1983     }
1984   }
1985 }
1986 \l_tmpa_tl
1987 }
1988 }
1989
1990 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1991   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1992     { \tl_tail:N \l_tmpa_tl }
1993   \tl_if_single:NTF \l_tmpa_tl {
1994     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1995       \exp_after:wN \str_set:Nn \exp_after:wN
1996         \l_stex_get_symbol_uri_str \l_tmpa_tl
1997     }{
1998       % TODO
1999       % tail is not a single group
2000     }
2001   }{
2002     % TODO
2003     % tail is not a single group
2004   }
2005 }

```

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

23.2 Notations

```

2006 <@@=stex_notation>
2007 notation arguments:
2008 \keys_define:nn { stex / notation } {
2009   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2009   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2010   prec .str_set_x:N = \l__stex_notation_prec_str ,
2011   op .tl_set:N = \l__stex_notation_op_tl ,
2012   unknown .code:n = \str_set:Nx
2013     \l__stex_notation_variant_str \l_keys_key_str
2014 }
2015
2016 \cs_new_protected:Nn \__stex_notation_args:n {
2017   \str_clear:N \l__stex_notation_lang_str
2018   \str_clear:N \l__stex_notation_variant_str

```

```

2019 \str_clear:N \l__stex_notation_prec_str
2020 \tl_clear:N \l__stex_notation_op_tl
2021
2022 \keys_set:nn { stex / notation } { #1 }
2023 }

```

\notation

```

2024 \NewDocumentCommand \notation { 0{ } m } {
2025   \__stex_notation_args:n { #1 }
2026   \tl_clear:N \l_stex_symdecl_definiens_tl
2027   \stex_get_symbol:n { #2 }
2028   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2029 }
2030 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

2031 \cs_new_protected:Nn \stex_notation_do:nn {
2032   \prop_set_eq:Nc \l_tmpa_prop {
2033     g_stex_symdecl_ #1 _prop
2034   }
2035
2036   \prop_clear:N \l_tmpb_prop
2037   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2038   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2039   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2040
2041   % precedences
2042   \seq_clear:N \l_tmpb_seq
2043   \exp_args:NNno
2044   \str_if_empty:NTF \l__stex_notation_prec_str {
2045     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2046     \int_compare:nNnTF \l_tmpa_str = 0 {
2047       \exp_args:NNnx
2048       \prop_put:Nno \l_tmpb_prop { opprec }
2049       { \neginfprec }
2050     }{
2051       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2052     }
2053   } {
2054     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2055       \exp_args:NNnx
2056       \prop_put:Nno \l_tmpb_prop { opprec }
2057       { \neginfprec }
2058       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2059       \int_step_inline:nn { \l_tmpa_str } {
2060         \exp_args:NNx
2061         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2062       }
2063     }{
2064       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2065       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2066         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2067         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2068         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2069         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2070         \seq_map_inline:Nn \l_tmpa_seq {
2071             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2072         }
2073     }
2074     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2075 }{
2076     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2077     \int_compare:nNnTF \l_tmpa_str = 0 {
2078         \exp_args:NNnx
2079         \prop_put:Nno \l_tmpb_prop { opprec }
2080         { \infprec }
2081     }{
2082         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2083     }
2084 }
2085 }
2086 }
2087
2088 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2089 \int_step_inline:nn { \l_tmpa_str } {
2090     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2091         \exp_args:NNx
2092         \seq_put_right:Nn \l_tmpb_seq {
2093             \prop_item:Nn \l_tmpb_prop { opprec }
2094         }
2095     }
2096 }
2097
2098 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2099 \tl_clear:N \l_tmpa_tl
2100
2101 \int_compare:nNnTF \l_tmpa_str = 0 {
2102     \exp_args:NNe
2103     \cs_set:Npn \l__stex_notation_macrocode_cs {
2104         \_stex_term_math_oms:nnnn { #1 }
2105         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2106         { \prop_item:Nn \l_tmpb_prop { opprec } }
2107         { \exp_not:n { #2 } }
2108     }
2109     \__stex_notation_final:
2110 }{
2111     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2112     \str_if_in:NnTF \l_tmpb_str b {
2113         \exp_args:Nne \use:nn
2114         {
2115             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2116             \cs_set:Npn \l_tmpa_str { {
2117                 \_stex_term_math_omb:nnnn { #1 }
2118                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2119                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2120                 { \exp_not:n { #2 } }
2121             }}

```

```

2122   }{
2123     \str_if_in:NnTF \l_tmpb_str B {
2124       \exp_args:Nne \use:nn
2125       {
2126         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2127         \cs_set:Npn \l_tmpa_str } { {
2128           \stex_term_math_omb:nnnn { #1 }
2129           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2130           { \prop_item:Nn \l_tmpb_prop { opprec } }
2131           { \exp_not:n { #2 } }
2132         } }
2133       }{
2134         \exp_args:Nne \use:nn
2135         {
2136           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2137           \cs_set:Npn \l_tmpa_str } { {
2138             \stex_term_math_oma:nnnn { #1 }
2139             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2140             { \prop_item:Nn \l_tmpb_prop { opprec } }
2141             { \exp_not:n { #2 } }
2142           } }
2143       }
2144     }
2145
2146     \int_zero:N \l_tmpa_int
2147     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2148     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2149     \__stex_notation_arguments:
2150   }
2151 }

```

(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

```

2152 \cs_new_protected:Nn \__stex_notation_arguments: {
2153   \int_incr:N \l_tmpa_int
2154   \str_if_empty:NnTF \l_tmpa_str {
2155     \__stex_notation_final:
2156   }{
2157     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2158     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2159     \str_if_eq:VnTF \l_tmpb_str a {
2160       \__stex_notation_argument_assoc:n
2161     }{
2162       \str_if_eq:VnTF \l_tmpb_str B {
2163         \__stex_notation_argument_assoc:n
2164       }{
2165         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2166         \tl_put_right:Nx \l_tmpa_tl {
2167           { \stex_term_math_arg:nnn
2168             { \int_use:N \l_tmpa_int }
2169             { \l_tmpb_str }
2170             { ####\int_use:N \l_tmpa_int }
2171           }

```

```

2172     }
2173     \__stex_notation_arguments:
2174   }
2175 }
2176 }
2177 }

```

(End definition for __stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2178 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2179   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2180   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2181   \tl_put_right:Nx \l_tmpa_tl {
2182     { \_stex_term_math_assoc_arg:nnnn
2183       { \int_use:N \l_tmpa_int }
2184       { \l_tmpb_str }
2185       \exp_args:No \exp_not:n
2186       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2187       { ####\int_use:N \l_tmpa_int }
2188     }
2189   }
2190   \__stex_notation_arguments:
2191 }

```

(End definition for _stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2192 \cs_new_protected:Nn \__stex_notation_final: {
2193   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2194   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2195   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2196   \exp_args:Nne \use:nn
2197   {
2198     \cs_generate_from_arg_count:cNnn {
2199       stex_notation_ \l_tmpa_str \c_hash_str
2200       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2201       _cs
2202     }
2203     \cs_gset:Npn \l_tmpb_str { { {
2204       \exp_after:wN \exp_after:wN \exp_after:wN
2205       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2206       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2207     } } }
2208
2209     \tl_if_empty:NF \l__stex_notation_op_tl {
2210       \cs_gset:cpx {
2211         stex_op_notation_ \l_tmpa_str \c_hash_str
2212         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2213         _cs
2214       } {
2215         \_stex_term_oms:nnn {
2216           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2217           \l__stex_notation_lang_str

```

```

2218     }{
2219         \l_tmpa_str
2220     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2221 }
2222 }
2223
2224
2225
2226 \stex_debug:nn{symbols}{
2227     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2228     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2229     Operator~precedence:~
2230     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2231     Argument~precedences:~
2232     \seq_use:Nn \l_tmpa_seq {,~}^^J
2233     Notation: \cs_meaning:c {
2234         stex_notation_ \l_tmpa_str \c_hash_str
2235         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2236         _cs
2237     }
2238 }
2239
2240 \prop_gset_eq:cN {
2241     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2242     \c_hash_str \l__stex_notation_lang_str _prop
2243 } \l_tmpb_prop
2244
2245 \exp_args:Nx
2246 \stex_add_to_current_module:n {
2247     \prop_get:cnN {
2248         g_stex_symdecl_
2249         \prop_item:Nn \l_tmpb_prop { symbol }
2250         _prop
2251     } { notations } \exp_not:N \l_tmpa_seq
2252     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2253         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2254     }
2255     \prop_put:cno {
2256         g_stex_symdecl_
2257         \prop_item:Nn \l_tmpb_prop { symbol }
2258         _prop
2259     } { notations } \exp_not:N \l_tmpa_seq
2260 }
2261
2262 \stex_if_smsmode:TF {
2263     \stex_smsmode_set_codes:
2264     \exp_args:Nx \stex_add_to_sms:n {
2265         \prop_gset_from_keyval:cn {
2266             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2267             \c_hash_str \l__stex_notation_lang_str _prop
2268         } {
2269             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2270             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2271             variant     = \prop_item:Nn \l_tmpb_prop { variant }      ,

```

```

2272         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2273         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2274     }
2275 }
2276 }{
2277   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2278   \seq_put_right:Nx \l_tmpa_seq {
2279     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2280   }
2281   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2282   \prop_set_eq:cN {
2283     g_stex_symdecl_ \l_tmpa_str _prop
2284   } \l_tmpa_prop
2285
2286   % HTML annotations
2287   \stex_if_do_html:T {
2288     \stex_annotate_invisible:nnn { notation }
2289     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2290       \stex_annotate_invisible:nnn { notationfragment }
2291       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2292       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2293       \stex_annotate_invisible:nnn { precedence }
2294       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2295         \seq_use:Nn \l_tmpa_seq { x }
2296       }{}
2297
2298       \int_zero:N \l_tmpa_int
2299       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2300       \tl_clear:N \l_tmpa_tl
2301       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2302         \int_incr:N \l_tmpa_int
2303         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2304         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2305         \str_if_eq:VnTF \l_tmpb_str a {
2306           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2307             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2308             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2309           } }
2310         }{
2311           \str_if_eq:VnTF \l_tmpb_str B {
2312             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2313               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2314               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2315             } }
2316           }{
2317             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2318               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2319             } }
2320           }
2321         }
2322       }
2323       \stex_annotate_invisible:nnn { notationcomp }{}{
2324         $ \exp_args:Nno \use:nn { \use:c {
2325           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```



```

2326         \c_hash_str \l__stex_notation_variant_str
2327         \c_hash_str \l__stex_notation_lang_str _cs
2328     } } { \l_tmpa_tl } $
2329 }
2330 }
2331 }
2332 }
2333 }

```

(End definition for `__stex_notation_final:`.)

\symdef

```

2334 \keys_define:nn { stex / symdef } {
2335   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2336   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2337   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2338   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2339   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2340   op        .tl_set:N   = \l__stex_notation_op_tl ,
2341   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2342   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2343   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2344   unknown   .code:n     = \str_set:Nx
2345             \l__stex_notation_variant_str \l_keys_key_str
2346 }
2347
2348 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2349   \str_clear:N \l_stex_symdecl_name_str
2350   \str_clear:N \l_stex_symdecl_args_str
2351   \bool_set_false:N \l_stex_symdecl_local_bool
2352   \tl_clear:N \l_stex_symdecl_type_tl
2353   \tl_clear:N \l_stex_symdecl_definiens_tl
2354   \str_clear:N \l__stex_notation_lang_str
2355   \str_clear:N \l__stex_notation_variant_str
2356   \str_clear:N \l__stex_notation_prec_str
2357   \tl_clear:N \l__stex_notation_op_tl
2358
2359   \keys_set:nn { stex / symdef } { #1 }
2360 }
2361
2362 \NewDocumentCommand \symdef { 0{} m } {
2363   \__stex_notation_symdef_args:n { #1 }
2364   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2365   \stex_symdecl_do:n { #2 }
2366   \exp_args:Nx \stex_notation_do:nn {
2367     \prop_item:Nn \l_tmpa_prop { module } ?
2368     \prop_item:Nn \l_tmpa_prop { name }
2369   }
2370 }
2371 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2372 </package>

```

Chapter 24

STEX -Terms Implementation

```
2373 <*package>
2374
2375 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2376
2377 <@@=stex_terms>
2378
2379 Warnings and error messages
2380 \msg_new:nnn{stex}{error/nonotation}{
2381   Symbol~#1~invoked,~but~has~no~notation~#2!
2382 }
2383 \msg_new:nnn{stex}{error/notationarg}{
2384   Error~in~parsing~notation~#1
2385 }
2386 \msg_new:nnn{stex}{error/noop}{
2387   Symbol~#1~has~no~operator~notation~for~notation~#2
2388 }
```

24.1 Symbol Invocations

Arguments:

```
2388 \keys_define:nn { stex / terms } {
2389   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2390   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2391   unknown .code:n = \str_set:Nx
2392     \l__stex_terms_variant_str \l_keys_key_str
2393 }
2394
2395 \cs_new_protected:Nn \__stex_terms_args:n {
2396   \str_clear:N \l__stex_terms_lang_str
2397   \str_clear:N \l__stex_terms_variant_str
2398   \str_clear:N \l__stex_terms_prec_str
2399   \tl_clear:N \l__stex_terms_op_tl
2400
2401   \keys_set:nn { stex / terms } { #1 }
```

2402 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2403 \cs_new_protected:Nn \stex_invoke_symbol:n {
2404   \if_mode_math:
2405     \exp_after:wN \__stex_terms_invoke_math:n
2406   \else:
2407     \exp_after:wN \__stex_terms_invoke_text:n
2408   \fi: { #1 }
2409 }
```

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

__stex_terms_invoke_math:n

```
2410 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2411   \peek_charcode_remove:NTF ! {
2412     \peek_charcode:NTF [ {
2413       \__stex_terms_invoke_op:nw { #1 }
2414     }{
2415       \peek_charcode_remove:NTF ! {
2416         \peek_charcode:NTF [ {
2417           \__stex_terms_invoke_op_custom:nw
2418         }{
2419           % TODO throw error
2420         }
2421       }{
2422         \__stex_terms_invoke_op:nw { #1 } []
2423       }
2424     }{
2425       \peek_charcode_remove:NTF * {
2426         \__stex_terms_invoke_text:n { #1 }
2427       }{
2428         \peek_charcode:NTF [ {
2429           \__stex_terms_invoke_math:nw { #1 }
2430         }{
2431           \__stex_terms_invoke_math:nw { #1 } []
2432         }
2433       }
2434     }
2435   }
2436 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2437 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2438   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2439     \stex_highlight_term:nn{#1}{#2}
2440   }
2441 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2442 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2443   \_stex_terms_args:n { #2 }
2444   \cs_if_exist:cTF {
2445     stex_op_notation_ #1 \c_hash_str
2446     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2447   }{
2448     \csname stex_op_notation_ #1 \c_hash_str
2449       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2450     \endcsname
2451   }{
2452     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2453   }
2454 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2455 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2456   \_stex_terms_args:n { #2 }
2457   \prop_set_eq:Nc \l_tmpa_prop {
2458     g_stex_symdecl_ #1 _prop
2459   }
2460   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2461   \seq_if_empty:NTF \l_tmpa_seq {
2462     \msg_error:nnxn{stex}{error/nonotation}{#1}{s}
2463   } {
2464     \seq_if_in:NxTF \l_tmpa_seq
2465     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2466       \use:c{
2467         stex_notation_ #1 \c_hash_str
2468         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2469         _cs
2470       }
2471     }{
2472       \str_if_empty:NTF \l__stex_terms_variant_str {
2473         \str_if_empty:NTF \l__stex_terms_lang_str {
2474           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2475           \use:c{
2476             stex_notation_ #1 \c_hash_str \l_tmpa_str
2477             _cs
2478           }
2479         }{
2480           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2481             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2482           }
2483         }
2484       }{
2485         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2486           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2487         }
2488       }
2489     }
2490 }

```

```
2491 }
(End definition for \_stex_terms_invoke_math:nw.)
```

```
\_stex_terms_invoke_text:n
2492 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2493   \peek_charcode_remove:NTF ! {
2494     \stex_term_custom:nn { #1 } { }
2495   }{
2496     \prop_set_eq:Nc \l_tmpa_prop {
2497       g_stex_symdecl_ #1 _prop
2498     }
2499     \prop_get:Nn \l_tmpa_prop { args } \l_tmpa_str
2500     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2501   }
2502 }
(End definition for \_stex_terms_invoke_text:n.)
```

24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2503 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2504 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2505 \int_new:N \l__stex_terms_downprec
2506 \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
2507 \tl_set:Nn \l__stex_terms_left_bracket_str (
2508 \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

```
2509 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2510   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2511     \bool_set_false:N \l__stex_terms_brackets_done_bool
2512     #2
2513   } {
2514     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2515       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2516         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2517         \dobrackets { #2 }
2518       }
2519     }{ #2 }
2520   }
2521 }
```

(End definition for `_stex_terms_maybe_brackets:nn`.)

\dobrackets

```
2522 \bool_new:N \l__stex_terms_brackets_done_bool
2523 %\RequirePackage{scalerel}
2524 \cs_new_protected:Npn \dobrackets #1 {
2525   %\ThisStyle{\if D\m@switch
2526   %   \exp_args:Nnx \use:nn
2527   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2528   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2529   % \else
2530   \exp_args:Nnx \use:nn
2531   {
2532     \bool_set_true:N \l__stex_terms_brackets_done_bool
2533     \int_set:Nn \l__stex_terms_downprec \infpref
2534     \l__stex_terms_left_bracket_str
2535     #1
2536   }
2537   {
2538     \bool_set_false:N \l__stex_terms_brackets_done_bool
2539     \l__stex_terms_right_bracket_str
2540     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2541   }
2542   %\fi}
2543 }
```

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

\withbrackets

```
2544 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2545   \exp_args:Nnx \use:nn
2546   {
2547     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2548     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2549     #3
2550   }
2551   {
2552     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2553     {\l__stex_terms_left_bracket_str}
2554     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2555     {\l__stex_terms_right_bracket_str}
2556   }
2557 }
```

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

\STEXinvisible

```
2558 \cs_new_protected:Npn \STEXinvisible #1 {
2559   \stex_annotate_invisible:n { #1 }
2560 }
```

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

OMDoc terms:

_stex_term_math_oms:nnnn

```
2561 \cs_new_protected:Nn \_stex_term_oms:nnn {
2562   \stex_annotate:nnn{ OMID }{ #2 }{
2563     \stex_highlight_term:nn { #1 } { #3 }
2564   }
2565 }
2566
2567 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2568   \__stex_terms_maybe_brackets:nn { #3 }{
2569     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2570   }
2571 }
```

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

_stex_term_math_oma:nnnn

```
2572 \cs_new_protected:Nn \_stex_term_oma:nnn {
2573   \stex_annotate:nnn{ OMA }{ #2 }{
2574     \stex_highlight_term:nn { #1 } { #3 }
2575   }
2576 }
2577
2578 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2579   \__stex_terms_maybe_brackets:nn { #3 }{
2580     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2581   }
2582 }
```

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

_stex_term_math_omb:nnnn

```
2583 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2584   \stex_annotate:nnn{ OMBIND }{ #2 }{
2585     \stex_highlight_term:nn { #1 } { #3 }
2586   }
2587 }
2588
2589 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2590   \__stex_terms_maybe_brackets:nn { #3 }{
2591     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2592   }
2593 }
```

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

_stex_term_math_arg:nnn

```
2594 \cs_new_protected:Nn \_stex_term_arg:nn {
2595   \stex_unhighlight_term:n {
2596     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2597   }
2598 }
2599 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2600   \exp_args:Nnx \use:nn
2601   { \int_set:Nn \l__stex_terms_downprec { #2 } }
```

```

2602     \stex_term_arg:nn { #1 }{ #3 }
2603   }
2604   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2605 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2606 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2607   \clist_set:Nn \l_tmpa_clist{ #4 }
2608   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2609     \tl_set:Nn \l_tmpa_tl { #4 }
2610   }{
2611     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2612     \clist_reverse:N \l_tmpa_clist
2613     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2614
2615     \clist_map_inline:Nn \l_tmpa_clist {
2616       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2617         \exp_args:Nno
2618         \l_tmpa_cs { ##1 } \l_tmpa_tl
2619       }
2620     }
2621
2622   }
2623   \exp_args:Nnno
2624   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2625 }

```

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

`\stex_term_custom:nn`

```

2626 \cs_new_protected:Nn \stex_term_custom:nn {
2627   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2628   \str_set:Nn \l_tmpa_str { #2 }
2629   \tl_clear:N \l_tmpa_tl
2630   \int_zero:N \l_tmpa_int
2631   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2632   \__stex_terms_custom_loop:
2633 }

```

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

`__stex_terms_custom_loop:`

```

2634 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2635   \bool_set_false:N \l_tmpa_bool
2636   \bool_while_do:nn {
2637     \str_if_eq_p:ee X {
2638       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2639     }
2640   }{
2641     \int_incr:N \l_tmpa_int
2642   }
2643
2644   \peek_charcode:NNTF [ {

```



```

2645 % notation/text component
2646 \__stex_terms_custom_component:w
2647 } {
2648 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2649 % all arguments read => finish
2650 \__stex_terms_custom_final:
2651 } {
2652 % arguments missing
2653 \peek_charcode_remove:NTF * {
2654 % invisible, specific argument position or both
2655 \peek_charcode:NTF [ {
2656 % visible specific argument position
2657 \__stex_terms_custom_arg:wn
2658 } {
2659 % invisible
2660 \peek_charcode_remove:NTF * {
2661 % invisible specific argument position
2662 \__stex_terms_custom_arg_inv:wn
2663 } {
2664 % invisible next argument
2665 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2666 }
2667 }
2668 } {
2669 % next normal argument
2670 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2671 }
2672 }
2673 }
2674 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2675 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2676 \bool_set_true:N \l_tmpa_bool
2677 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2678 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2679 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2680 \str_set:Nx \l_tmpb_str {
2681 \str_item:Nn \l_tmpa_str { #1 }
2682 }
2683 \str_case:VnTF \l_tmpb_str {
2684 { X } {
2685 \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2686 }
2687 { i } { \__stex_terms_custom_set_X:n { #1 } }
2688 { b } { \__stex_terms_custom_set_X:n { #1 } }
2689 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2690 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2691 }{}{

```

```

2692 \msg_error:nxx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2693 }
2694
2695 \bool_if:nTF \l_tmpa_bool {
2696   \tl_put_right:Nx \l_tmpa_tl {
2697     \stex_annotate_invisible:n {
2698       \stex_term_arg:nn { \int_eval:n { #1 } }
2699       \exp_not:n { { #2 } }
2700     }
2701   }
2702 } {
2703   \tl_put_right:Nx \l_tmpa_tl {
2704     \stex_term_arg:nn { \int_eval:n { #1 } }
2705     \exp_not:n { { #2 } }
2706   }
2707 }
2708
2709 \__stex_terms_custom_loop:
2710 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2711 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2712   \str_set:Nx \l_tmpa_str {
2713     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2714     X
2715     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2716   }
2717 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2718 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2719   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2720   \__stex_terms_custom_loop:
2721 }

```

(End definition for __stex_terms_custom_component:.)

__stex_terms_custom_final:

```

2722 \cs_new_protected:Nn \__stex_terms_custom_final: {
2723   \int_compare:nNnTF \l_tmpb_int = 0 {
2724     \exp_args:Nnno \stex_term_oms:nnn
2725   } {
2726     \str_if_in:NnTF \l_tmpa_str {b} {
2727       \exp_args:Nnno \stex_term_ombind:nnn
2728     } {
2729       \exp_args:Nnno \stex_term_oma:nnn
2730     }
2731   }
2732   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2733 }

```

(End definition for `_stex_terms_custom_final:`.)

```

\symref
\symname 2734 \NewDocumentCommand \symref { m m }{
2735   \let\compemph_uri_prev:\compemph@uri
2736   \let\compemph@uri\symrefemph@uri
2737   \STEXsymbol{#1}![#2]
2738   \let\compemph@uri\compemph_uri_prev:
2739 }
2740
2741 \keys_define:nn { stex / symname } {
2742   post      .str_set_x:N = \l_stex_symname_post_str
2743 }
2744
2745 \cs_new_protected:Nn \stex_symname_args:n {
2746   \str_clear:N \l_stex_symname_post_str
2747   \keys_set:nn { stex / symname } { #1 }
2748 }
2749
2750 \NewDocumentCommand \symname { 0{} m }{
2751   \stex_symname_args:n { #1 }
2752   \stex_get_symbol:n { #2 }
2753   \str_set:Nx \l_tmpa_str {
2754     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2755   }
2756   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2757
2758   \let\compemph_uri_prev:\compemph@uri
2759   \let\compemph@uri\symrefemph@uri
2760   \exp_args:NNx \use:nn
2761   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2762     \l_tmpa_str \l_stex_symname_post_str
2763   ] }
2764   \let\compemph@uri\compemph_uri_prev:
2765 }

```

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

24.3 Notation Components

2766 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2767
2768 \str_new:N \l__stex_notationcomps_highlight_uri_str
2769 \cs_new_protected:Nn \stex_highlight_term:nn {
2770   \exp_args:Nnx
2771   \use:nn {
2772     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2773     #2
2774   } {
2775     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2776     { \l__stex_notationcomps_highlight_uri_str }
2777   }

```

```

2778 }
2779
2780 \cs_new_protected:Nn \stex_unhighlight_term:n {
2781 % \latexml_if:TF {
2782 %   #1
2783 % } {
2784 %   \rustex_if:TF {
2785 %     #1
2786 %   } {
2787     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2788 %   }
2789 % }
2790 }

```

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

```

\comp
\compemph@uri 2791 \cs_new_protected:Npn \comp #1 {
\compemph 2792 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2793 \rustex_if:TF {
\defemph@uri 2794 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2795 }{
\symrefemph@uri 2796 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2797 }
2798 }
2799 }
2800
2801 \cs_new_protected:Npn \compemph@uri #1 #2 {
2802 \compemph{ #1 }
2803 }
2804
2805
2806 \cs_new_protected:Npn \compemph #1 {
2807 \textcolor{blue}{#1}
2808 }
2809
2810 \cs_new_protected:Npn \defemph@uri #1 #2 {
2811 \defemph{#1}
2812 }
2813
2814 \cs_new_protected:Npn \defemph #1 {
2815 \textbf{#1}
2816 }
2817
2818 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2819 \symrefemph{#1}
2820 }
2821
2822 \cs_new_protected:Npn \symrefemph #1 {
2823 \textbf{#1}
2824 }

```

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

\ellipses

```
2825 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2826 \bool_new:N \l_stex_inarray_bool
\parrayline 2827 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2828 \NewDocumentCommand \parray { m m } {
\parraycell 2829 \begingroup
2830 \bool_set_true:N \l_stex_inarray_bool
2831 \begin{array}{#1}
2832 #2
2833 \end{array}
2834 \endgroup
2835 }
2836
2837 \NewDocumentCommand \prmatrix { m } {
2838 \begingroup
2839 \bool_set_true:N \l_stex_inarray_bool
2840 \begin{matrix}
2841 #1
2842 \end{matrix}
2843 \endgroup
2844 }
2845
2846 \def \maybepline {
2847 \bool_if:NT \l_stex_inarray_bool {\hline}
2848 }
2849
2850 \def \parrayline #1 #2 {
2851 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2852 }
2853
2854 \def \pmrow #1 { \parrayline{}{ #1 } }
2855
2856 \def \parraylineh #1 #2 {
2857 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2858 }
2859
2860 \def \parraycell #1 {
2861 #1 \bool_if:NT \l_stex_inarray_bool {&}
2862 }

```

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

```
2863 \endpackage
```

Chapter 25

STEX -Structural Features Implementation

```
2864 <*package>
2865
2866 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2867
2868 <@@=stex_features>
      Warnings and error messages
2869
```

25.1 The feature environment

structural@feature

```
2870
2871 \NewDocumentEnvironment{structural@feature}{ m m m }{
2872   \stex_if_in_module:F {
2873     \msg_set:nnn{stex}{error/nomodule}{
2874       Structural~Feature~has~to~occur~in~a~module:\\
2875       Feature~#2~of~type~#1\\
2876       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2877     }
2878     \msg_error:nn{stex}{error/nomodule}
2879   }
2880
2881   \str_set:Nx \l_stex_module_name_str {
2882     \prop_item:Nn \l_stex_current_module_prop
2883       { name } / #2 - feature
2884   }
2885
2886   \str_set:Nx \l_stex_module_ns_str {
2887     \prop_item:Nn \l_stex_current_module_prop
2888       { ns }
2889   }
2890
```

```

2891
2892 \str_clear:N \l_tmpa_str
2893 \seq_clear:N \l_tmpa_seq
2894 \tl_clear:N \l_tmpa_tl
2895 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2896   origname = #2,
2897   name     = \l_stex_module_name_str ,
2898   ns       = \l_stex_module_ns_str ,
2899   imports  = \exp_not:o { \l_tmpa_seq } ,
2900   constants = \exp_not:o { \l_tmpa_seq } ,
2901   content  = \exp_not:o { \l_tmpa_tl } ,
2902   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2903   lang     = \l_stex_module_lang_str ,
2904   sig      = \l_tmpa_str ,
2905   meta     = \l_tmpa_str ,
2906   feature  = #1 ,
2907 }
2908
2909 \stex_if_smsmode:TF {
2910   \stex_smsmode_set_codes:
2911 } {
2912   \begin{stex_annotate_env}{ feature:#1 }{}
2913   \stex_annotate_invisible:nnn{header}{}{ #3 }
2914 }
2915 }{
2916   \str_set:Nx \l_tmpa_str {
2917     c_stex_feature_
2918     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2919     \prop_item:Nn \l_stex_current_module_prop { name }
2920     _prop
2921   }
2922   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2923   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2924   \stex_if_smsmode:TF {
2925     \exp_args:Nx \stex_add_to_sms:n {
2926       \prop_gset_from_keyval:cn {
2927         c_stex_feature_
2928         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2929         \prop_item:Nn \l_stex_current_module_prop { name }
2930         _prop
2931       } {
2932         origname = #2,
2933         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2934         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2935         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2936         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2937         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2938         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2939         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2940         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2941         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2942         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2943       }
2944     }

```

```

2945 } {
2946     \end{stex_annotate_env}
2947 }
2948 }
2949

```

25.2 Features

structure

```

2950
2951 \prop_new:N \l_stex_all_structures_prop
2952
2953 \keys_define:nn { stex / features / structure } {
2954     name .str_set_x:N = \l__stex_features_structure_name_str ,
2955 }
2956
2957 \cs_new_protected:Nn \__stex_features_structure_args:n {
2958     \str_clear:N \l__stex_features_structure_name_str
2959     \keys_set:nn { stex / features / structure } { #1 }
2960 }
2961
2962 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2963 % \__stex_features_structure_args:n { ##1 }
2964 % \str_if_empty:NT \l__stex_features_structure_name_str {
2965 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2966 % }
2967 %} {
2968 %
2969 %}
2970
2971 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2972     \__stex_features_structure_args:n { #1 }
2973     \str_if_empty:NT \l__stex_features_structure_name_str {
2974         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2975     }
2976     \exp_args:Nnnx
2977     \begin{structural@feature}{ structure }
2978         { \l__stex_features_structure_name_str }{}
2979         \seq_clear:N \l_tmpa_seq
2980         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2981     }{
2982         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2983         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2984         \str_set:Nx \l_tmpa_str {
2985             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2986             \prop_item:Nn \l_stex_current_module_prop { name }
2987         }
2988         \seq_map_inline:Nn \l_tmpa_seq {
2989             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2990         }
2991         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2992         \exp_args:Nnx

```



```

2994 \AddToHookNext { env / mathstructure / after }{
2995 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2996 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2997 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2998 \STEXexport {
2999 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3000 {\prop_item:Nn \l_stex_current_module_prop { origname }}
3001 {\l_tmpa_str}
3002 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
3003 {#2}{\l_tmpa_str}
3004 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3005 % \prop_item:Nn \l_stex_current_module_prop { origname },
3006 % \l_tmpa_str
3007 % }
3008 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
3009 % #2,\l_tmpa_str
3010 % }
3011 % \tl_set:cx { #2 } {
3012 % \stex_invoke_structure:n { \l_tmpa_str }
3013 % }
3014 % }
3015
3016 \end{structural@feature}
3017 % \g_stex_last_feature_prop
3018 }

```

\instantiate

```

3019 \seq_new:N \l__stex_features_structure_field_seq
3020 \str_new:N \l__stex_features_structure_field_str
3021 \str_new:N \l__stex_features_structure_def_tl
3022 \prop_new:N \l__stex_features_structure_prop
3023 \NewDocumentCommand \instantiate { m O{} m }{
3024 \stex_smsmode_set_codes:
3025 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
3026 \prop_set_eq:Nc \l__stex_features_structure_prop {
3027 c_stex_feature_\l_tmpa_str _prop
3028 }
3029 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3030 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3031 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3032 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3033 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3034 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3035 {!} \l_tmpa_tl
3036 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3037 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3038 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3039 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3040 }{
3041 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3042 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3043 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3044 \l_tmpa_tl
3045 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

3046         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3047         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3048     }{
3049         \tl_clear:N \l_tmpb_tl
3050     }
3051 }
3052 }{
3053     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3054     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3055         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3056         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3057         \tl_clear:N \l_tmpa_tl
3058     }{
3059         % TODO throw error
3060     }
3061 }
3062 % \l_tmpa_str: name
3063 % \l_tmpa_tl: definiens
3064 % \l_tmpb_tl: notation
3065 \tl_if_empty:NT \l__stex_features_structure_field_str {
3066     % TODO throw error
3067 }
3068 \str_clear:N \l_tmpb_str
3069
3070 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3071 \seq_map_inline:Nn \l_tmpa_seq {
3072     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3073     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3074     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3075         \seq_map_break:n {
3076             \str_set:Nn \l_tmpb_str { ####1 }
3077         }
3078     }
3079 }
3080 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3081     \l_tmpb_str
3082
3083 \tl_if_empty:NTF \l_tmpb_tl {
3084     \tl_if_empty:NF \l_tmpa_tl {
3085         \exp_args:Nx \use:n {
3086             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3087         }
3088     }
3089 }{
3090     \tl_if_empty:NTF \l_tmpa_tl {
3091         \exp_args:Nx \use:n {
3092             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3093         }
3094     }
3095 }{
3096     \exp_args:Nx \use:n {
3097         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3098         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3099     }

```

```

3100     }
3101   }
3102   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3103   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3104   % #3/\l_stex_features_structure_field_str
3105   % \par
3106   % \expandafter\present\csname
3107   %   g_stex_symdecl_
3108   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3109   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3110   %   #3/\l_stex_features_structure_field_str
3111   %   _prop
3112   % \endcsname
3113 }
3114
3115 \tl_clear:N \l__stex_features_structure_def_tl
3116
3117 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3118 \seq_map_inline:Nn \l_tmpa_seq {
3119   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3120   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3121   \exp_args:Nx \use:n {
3122     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3123
3124     }
3125   }
3126
3127   \prop_if_exist:cF {
3128     g_stex_symdecl_
3129     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3130     \prop_item:Nn \l_stex_current_module_prop {name} ?
3131     #3/\l_tmpa_str
3132     _prop
3133   }{
3134     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3135     \l_tmpb_str
3136     \exp_args:Nx \use:n {
3137       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3138     }
3139   }
3140 }
3141
3142 \symdecl*[type={\STEXsymbol{module-type}}{
3143   \_stex_term_math_oms:nnnn {
3144     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3145     \prop_item:Nn \l__stex_features_structure_prop {name}
3146     }{}{0}{}
3147   }{}{#3}
3148
3149   % TODO: -> sms file
3150
3151   \tl_set:cx{ #3 }{
3152     \stex_invoke_structure:nnn {
3153       \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3154     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3155   } {
3156     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3157     \prop_item:Nn \l__stex_features_structure_prop {name}
3158   }
3159 }
3160
3161 }

```

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

\stex_invoke_structure:nnn

```

3162 % #1: URI of the instance
3163 % #2: URI of the instantiated module
3164 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3165   \tl_if_empty:nTF{ #3 }{
3166     \prop_set_eq:Nc \l__stex_features_structure_prop {
3167       c_stex_feature_ #2 _prop
3168     }
3169     \tl_clear:N \l_tmpa_tl
3170     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3171     \seq_map_inline:Nn \l_tmpa_seq {
3172       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3173       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3174       \cs_if_exist:cT {
3175         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3176       }{
3177         \tl_if_empty:NF \l_tmpa_tl {
3178           \tl_put_right:Nn \l_tmpa_tl {,}
3179         }
3180         \tl_put_right:Nx \l_tmpa_tl {
3181           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3182         }
3183       }
3184     }
3185     \exp_args:No \mathstrut \l_tmpa_tl
3186   }{
3187     \stex_invoke_symbol:n{#1/#3}
3188   }
3189 }

```

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

```

3190 </package>

```

Chapter 26

STEX -Statements Implementation

```
3191 <*package>
3192
3193 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3194
3195 \protected\def\ignorespacesandpars{
3196   \begingroup\catcode13=10\relax
3197   \@ifnextchar\par{
3198     \endgroup\expandafter\ignorespacesandpars\@gobble
3199   }{
3200     \endgroup
3201   }
3202 }
3203
3204 <@@=stex_statements>
3205
3206   Warnings and error messages
```

\titleemph

```
3206 \def\titleemph#1{\textbf{#1}}
```

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

26.1 Definitions

definiendum

```
3207 \keys_define:nn {stex / definiendum }{
3208   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3209   root      .str_set_x:N    = \l__stex_statements_definiendum_root_str,
3210   gfa       .str_set_x:N    = \l__stex_statements_definiendum_gfa_str
3211 }
3212 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3213   \str_clear:N \l__stex_statements_definiendum_root_str
3214   \tl_clear:N \l__stex_statements_definiendum_post_tl
3215   \str_clear:N \l__stex_statements_definiendum_gfa_str
```

```

3216 \keys_set:nn { stex / definiendum } { #1 }
3217 }
3218 \NewDocumentCommand \definiendum { 0{ } m m } {
3219   \__stex_statements_definiendum_args:n { #1 }
3220   \stex_get_symbol:n { #2 }
3221   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3222   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3223     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3224       \tl_set:Nn \l_tmpa_tl { #3 }
3225     } {
3226       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3227       \tl_set:Nn \l_tmpa_tl {
3228         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3229       }
3230     }
3231   } {
3232     \tl_set:Nn \l_tmpa_tl { #3 }
3233   }
3234
3235   % TODO root
3236   \rustex_if:TF {
3237     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3238   } {
3239     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3240   }
3241 }
3242 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3243 \NewDocumentCommand \definame { 0{ } m } {
3244   \__stex_statements_definiendum_args:n { #1 }
3245   % TODO: root
3246   \stex_get_symbol:n { #2 }
3247   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3248   \str_set:Nx \l_tmpa_str {
3249     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3250   }
3251   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3252   \rustex_if:TF {
3253     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3254       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3255     }
3256   } {
3257     \defemph@uri {
3258       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3259     } { \l_stex_get_symbol_uri_str }
3260   }
3261 }
3262 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3263
3264 \keys_define:nn {stex / sdefinition }{
3265   type      .str_set_x:N = \sdefinitiontype,
3266   id        .str_set_x:N = \sdefinitionid,
3267   title     .tl_set:N    = \sdefinitiontitle
3268 }
3269 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3270   \str_clear:N \sdefinitiontype
3271   \str_clear:N \sdefinitionid
3272   \tl_clear:N \sdefinitiontitle
3273   \keys_set:nn { stex / sdefinition }{ #1 }
3274 }
3275
3276 \NewDocumentEnvironment{sdefinition}{0{}}{
3277   \__stex_statements_sdefinition_args:n{ #1 }
3278   \stex_reactivate_macro:N \definiendum
3279   \stex_reactivate_macro:N \definame
3280   \stex_smsmode_set_codes:
3281   \clist_set:No \l_tmpa_clist \sdefinitiontype
3282   \tl_clear:N \l_tmpa_tl
3283   \clist_map_inline:Nn \l_tmpa_clist {
3284     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3285       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3286     }
3287   }
3288   \tl_if_empty:NTF \l_tmpa_tl {
3289     \__stex_statements_sdefinition_start:
3290   }{
3291     \l_tmpa_tl
3292   }
3293   \stex_ref_new_doc_target:n \sdefinitionid
3294   \stex_if_smsmode:F {
3295     \exp_args:Nnnx
3296     \begin{stex_annotate_env}{definition}{}
3297     \str_if_empty:NF \sdefinitiontype {
3298       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3299     }
3300   }
3301 }{
3302   \stex_if_smsmode:F {
3303     \end{stex_annotate_env}
3304   }
3305   \clist_set:No \l_tmpa_clist \sdefinitiontype
3306   \tl_clear:N \l_tmpa_tl
3307   \clist_map_inline:Nn \l_tmpa_clist {
3308     \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3309       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3310     }
3311   }
3312   \tl_if_empty:NTF \l_tmpa_tl {
3313     \__stex_statements_sdefinition_end:
3314   }{
3315     \l_tmpa_tl

```

```

3316 }
3317 }

```

`\stexpatchdefinition`

```

3318 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3319   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3320     ~(\sdefinitiontitle)
3321   }~}
3322 }
3323 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3324
3325 \newcommand\stexpatchdefinition[3] [] {
3326   \str_set:Nx \l_tmpa_str{ #1 }
3327   \str_if_empty:NTF \l_tmpa_str {
3328     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3329     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3330   }{
3331     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3332     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3333   }
3334 }

```

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

`\inlinedef inline:`

```

3335 \NewDocumentCommand \inlinedef { m } {
3336   \begingroup
3337   \stex_reactivate_macro:N \definiendum
3338   \stex_reactivate_macro:N \definame
3339   \stex_ref_new_doc_target:n{
3340     #1
3341   }
3342 }

```

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

26.2 Assertions

`sassertion`

```

3343
3344 \keys_define:nn {stex / sassertion }{
3345   type      .str_set_x:N = \sassertiontype,
3346   id        .str_set_x:N = \sassertionid,
3347   title     .tl_set:N     = \sassertiontitle ,
3348   name      .str_set_x:N = \sassertionname
3349 }
3350 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3351   \str_clear:N \sassertiontype
3352   \str_clear:N \sassertionid
3353   \str_clear:N \sassertionname
3354   \tl_clear:N \sassertiontitle
3355   \keys_set:nn { stex / sassertion }{ #1 }
3356 }

```



```

3357
3358 \tl_new:N \g__stex_statements_aftergroup_tl
3359
3360 \NewDocumentEnvironment{sassertion}{0{}}{
3361   \__stex_statements_sassertion_args:n{ #1 }
3362   \stex_smsmode_set_codes:
3363   \clist_set:No \l_tmpa_clist \sassertiontype
3364   \tl_clear:N \l_tmpa_tl
3365   \clist_map_inline:Nn \l_tmpa_clist {
3366     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3367       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3368     }
3369   }
3370   \tl_if_empty:NTF \l_tmpa_tl {
3371     \__stex_statements_sassertion_start:
3372   }{
3373     \l_tmpa_tl
3374   }
3375   \stex_ref_new_doc_target:n \sassertionid
3376   \stex_if_smsmode:F {
3377     \exp_args:Nnnx
3378     \begin{stex_annotate_env}{assertion}{}
3379     \str_if_empty:NF \sassertiontype {
3380       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3381     }
3382   }
3383   {
3384     \stex_if_smsmode:F {
3385       \end{stex_annotate_env}
3386     }
3387     \clist_set:No \l_tmpa_clist \sassertiontype
3388     \tl_clear:N \l_tmpa_tl
3389     \clist_map_inline:Nn \l_tmpa_clist {
3390       \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3391         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3392       }
3393     }
3394     \tl_if_empty:NTF \l_tmpa_tl {
3395       \__stex_statements_sassertion_end:
3396     }{
3397       \l_tmpa_tl
3398     }
3399     \str_if_empty:NF \sassertionname {
3400       \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3401         \symdecl*{\sassertionname}
3402       }
3403       \aftergroup\g__stex_statements_aftergroup_tl
3404     }
3405   }

```

\stexpatchassertion

```

3406
3407 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3408   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3409     (\sassertiontitle)
3410   }~}
3411 }
3412 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3413
3414 \newcommand\stexpatchassertion[3] [] {
3415   \str_set:Nx \l_tmpa_str{ #1 }
3416   \str_if_empty:NTF \l_tmpa_str {
3417     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3418     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3419   }{
3420     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3421     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3422   }
3423 }

```

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

\inlineass inline:

```

3424 \NewDocumentCommand \inlineass { m } {
3425   \begingroup
3426   \stex_ref_new_doc_target:n{
3427     #1
3428   \endgroup
3429 }

```

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

26.3 Examples

sexample

```

3430
3431 \keys_define:nn {stex / sexample }{
3432   type      .str_set_x:N = \exampletype,
3433   id        .str_set_x:N = \sexampleid,
3434   title     .tl_set:N = \sexampletile,
3435   for       .clist_set:N = \sexamplefor,
3436 }
3437 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3438   \str_clear:N \sexampletype
3439   \str_clear:N \sexampleid
3440   \tl_clear:N \sexampletile
3441   \clist_clear:N \sexamplefor
3442   \keys_set:nn { stex / sexample }{ #1 }
3443 }
3444
3445 \NewDocumentEnvironment{sexample}{0{}}{
3446   \__stex_statements_sexample_args:n{ #1 }
3447   \stex_smsmode_set_codes:
3448   \clist_set:N \l_tmpa_clist \sexampletype
3449   \tl_clear:N \l_tmpa_tl
3450   \clist_map_inline:Nn \l_tmpa_clist {
3451     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3452     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3453   }
3454 }
3455 \tl_if_empty:NTF \l_tmpa_tl {
3456   \__stex_statements_sexample_start:
3457 }{
3458   \l_tmpa_tl
3459 }
3460 \stex_ref_new_doc_target:n \sexampleid
3461 \stex_if_smsmode:F {
3462   \seq_clear:N \l_tmpa_seq
3463   \clist_map_inline:Nn \sexamplefor {
3464     \str_if_eq:nnF{ ##1 }{}{
3465       \stex_get_symbol:n { ##1 }
3466       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3467         \l_stex_get_symbol_uri_str
3468       }
3469     }
3470   }
3471   \exp_args:Nnnx
3472   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3473   \str_if_empty:NF \sexamplotype {
3474     \stex_annotate_invisible:nnn{type}{\sexampletype}{ }
3475   }
3476 }
3477 }{
3478   \stex_if_smsmode:F {
3479     \end{stex_annotate_env}
3480   }
3481   \clist_set:Nn \l_tmpa_clist \sexamplotype
3482   \tl_clear:N \l_tmpa_tl
3483   \clist_map_inline:Nn \l_tmpa_clist {
3484     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3485       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3486     }
3487   }
3488   \tl_if_empty:NTF \l_tmpa_tl {
3489     \__stex_statements_sexample_end:
3490   }{
3491     \l_tmpa_tl
3492   }
3493 }

```

\stexpatchexample

```

3494
3495 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3496   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3497     (\sexamplotype)
3498   }~}
3499 }
3500 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3501
3502 \newcommand\stexpatchexample[3][ ] {
3503   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3504 \str_if_empty:NTF \l_tmpa_str {
3505   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3506   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3507 }{
3508   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3509   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3510 }
3511 }

```

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

`\inlineex` inline:

```

3512 \NewDocumentCommand \inlineex { m } {
3513   \begingroup
3514   \stex_ref_new_doc_target:n{
3515     #1
3516   }
3517 }

```

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

26.4 Logical Paragraphs

`sparagraph`

```

3518 \keys_define:nn { stex / sparagraph } {
3519   id      .str_set_x:N = \sparagraphid ,
3520   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3521   type    .str_set_x:N = \sparagraphtype ,
3522   for     .str_set_x:N = \sparagraphfor ,
3523   from    .tl_set_x:N  = \sparagraphfrom ,
3524   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
3525   name    .str_set:N   = \sparagraphname
3526 }
3527
3528 \cs_new_protected:Nn \stex_sparagraph_args:n {
3529   \tl_clear:N \l_stex_sparagraph_title_tl
3530   \tl_clear:N \sparagraphfrom
3531   \tl_clear:N \l_stex_sparagraph_start_tl
3532   \str_clear:N \sparagraphid
3533   \str_clear:N \sparagraphtype
3534   \str_clear:N \sparagraphfor
3535   \str_clear:N \sparagraphname
3536   \keys_set:nn { stex / sparagraph } { #1 }
3537 }
3538 \newif\if@in@omtext\@in@omtextfalse
3539
3540 \NewDocumentEnvironment {sparagraph} { 0{} } {
3541   \stex_sparagraph_args:n { #1 }
3542   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3543     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3544   }{
3545     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3546   }

```

```

3547 \@in@omtexttrue
3548 \stex_smsmode_set_codes:
3549 \clist_set:No \l_tmpa_clist \sparagraphtype
3550 \tl_clear:N \l_tmpa_tl
3551 \clist_map_inline:Nn \l_tmpa_clist {
3552   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3553     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3554   }
3555 }
3556 \tl_if_empty:NTF \l_tmpa_tl {
3557   \__stex_statements_sparagraph_start:
3558 }{
3559   \l_tmpa_tl
3560 }
3561 \stex_ref_new_doc_target:n \sparagraphid
3562 \stex_if_smsmode:F {
3563   \exp_args:Nnnx
3564   \begin{stex_annotate_env}{paragraph}{}
3565   \str_if_empty:NF \sparagraphtype {
3566     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
3567   }
3568 }
3569 \ignorespacesandpars
3570 }{
3571   \stex_if_smsmode:F {
3572     \end{stex_annotate_env}
3573   }
3574   \clist_set:No \l_tmpa_clist \sparagraphtype
3575   \tl_clear:N \l_tmpa_tl
3576   \clist_map_inline:Nn \l_tmpa_clist {
3577     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
3578       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
3579     }
3580   }
3581   \tl_if_empty:NTF \l_tmpa_tl {
3582     \__stex_statements_sparagraph_end:
3583   }{
3584     \l_tmpa_tl
3585   }
3586   \str_if_empty:NF \sparagraphname {
3587     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3588       \symdecl*{\sparagraphname}
3589     }
3590     \aftergroup\g__stex_statements_aftergroup_tl
3591   }
3592 }

```

\stexpatchparagraph

```

3593
3594 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
3595   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3596     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
3597       \titleemph{\l_stex_sparagraph_title_tl}:~
3598     }

```

```

3599   }{
3600   \titleemph{\l_stex_sparagraph_start_tl}~
3601   }
3602 }
3603 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
3604
3605 \newcommand\stexpatchparagraph[3] [] {
3606   \str_set:Nx \l_tmpa_str{ #1 }
3607   \str_if_empty:NTF \l_tmpa_str {
3608     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
3609     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
3610   }{
3611     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
3612     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
3613   }
3614 }

```

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

symboldoc

```

3615 \NewDocumentEnvironment{symboldoc}{m}{
3616   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3617   \seq_clear:N \l_tmpb_seq
3618   \seq_map_inline:Nn \l_tmpa_seq {
3619     \str_if_eq:nnF{ ##1 }{}{
3620       \stex_get_symbol:n { ##1 }
3621       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3622         \l_stex_get_symbol_uri_str
3623       }
3624     }
3625   }
3626   \par
3627   \exp_args:Nnnx
3628   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3629 }{
3630   \end{stex_annotate_env}
3631 }
3632 \</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

```
3633 <*package>
3634 <@@=stex_sproof>
3635
3636 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3637
```

27.2 Proofs

We first define some keys for the proof environment.

```
3638 \keys_define:nn { stex / spf } {
3639   id          .str_set:N = \l__stex_sproof_spf_id_str,
3640   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3641   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3642   from       .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3643   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3644   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3645   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3646   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3647   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3648   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3649 }
3650 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3651   \str_clear:N \l__stex_sproof_spf_id_str
3652   \tl_clear:N \l__stex_sproof_spf_display_tl
3653   \tl_clear:N \l__stex_sproof_spf_for_tl
3654   \tl_clear:N \l__stex_sproof_spf_from_tl
3655   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3656   \tl_clear:N \l__stex_sproof_spf_type_tl
3657   \tl_clear:N \l__stex_sproof_spf_title_tl

```

¹⁰EDNOTE: need an implementation for L^AT_EX_ML

```

3658 \tl_clear:N \l__stex_sproof_spf_continues_tl
3659 \tl_clear:N \l__stex_sproof_spf_functions_tl
3660 \tl_clear:N \l__stex_sproof_spf_method_tl
3661 \keys_set:nn { stex / spf }{ #1 }
3662 }

```

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

```

3663 \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.

```

3664 \newcount\count_ten
3665 \newenvironment{pst@with@label}[1]{
3666   \edef\pst@label{#1}
3667   \advance\count_ten by 1\relax
3668   \count_ten=1
3669 }{
3670   \advance\count_ten by -1\relax
3671 }

```

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

```

3672 \def\the@pst@label{
3673   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3674 }

```

(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.

```

3675 \keys_define:nn { stex / pstlabel }{
3676   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3677   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3678   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3679 }
3680 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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


```

3681 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3682 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3683 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3684 }
3685 \__stex_sproof_pstlabel_args:n {}
3686 \newcommand\setpstlabelstyle[1]{
3687   \__stex_sproof_pstlabel_args:n {#1}
3688 }
3689 \newcommand\setpstlabelstyledefault{%
3690   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3691 }

```

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

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

```

3692 \ExplSyntaxOff
3693 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3694 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3695 \def\pst@make@label@short#1#2{#2}
3696 \def\pst@make@label@empty#1#2{}
3697 \ExplSyntaxOn
3698 \def\pstlabelstyle#1{%
3699   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3700 }%
3701 \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.

```

3702 \def\next@pst@label{%
3703   \global\advance\count\count10 by 1%
3704 }%

```

(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

```

3705 \def\sproof@box{
3706   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3707 }
3708 \def\spf@proofend{\sproof@box}
3709 \def\sproofend{
3710   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3711     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3712   }
3713 }
3714 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3715 \def\spf@proofsketch@kw{Proof Sketch}
3716 \def\spf@proof@kw{Proof}
3717 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3718 \cs_if_exist:NT \bbl@loaded {
3719   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3720   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3721     \input{proof-ngerman.lda}
3722   }
3723   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3724     \input{proof-finnish.lda}
3725   }
3726   \clist_if_in:NnT \l_tmpa_clist {french}{
3727     \input{proof-french.lda}
3728   }
3729   \clist_if_in:NnT \l_tmpa_clist {russian}{
3730     \input{proof-russian.lda}
3731   }
3732 }
3733

```

spfsketch

```

3734 \newcommand\spfsketch[2][]{
3735   \__stex_sproof_spf_args:n{#1}
3736   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3737     \titleemph{
3738       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3739         \spf@proofsketch@kw
3740       }{
3741         \l__stex_sproof_spf_type_tl
3742       }
3743     }:
3744   }
3745   {-#2}
3746   %\sref@label@id{this \ifx\spf@type\empty\spf@proofsketch@kw\else\spf@type\fi}
3747   \sproofend
3748 }

```

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

EdN:11
EdN:12

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

```

3749 \newenvironment{spfeq}[2][]{
3750   \__stex_sproof_spf_args:n{#1}
3751   %\sref@target
3752   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3753     \titleemph{
3754       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3755         \spf@proof@kw
3756       }{
3757         \l__stex_sproof_spf_type_tl
3758       }
3759     }:

```

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

¹²EDNOTE: document above

```

3760 }
3761 {~#2}
3762 \begin{displaymath}\begin{array}{rcll}
3763 }{
3764 \end{array}\end{displaymath}
3765 }

```

(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.

```

3766 \newenvironment{spf@proof}[2][]{
3767   \__stex_sproof_spf_args:n{#1}
3768   %\sref@target
3769   \count_ten=10
3770   \par\noindent
3771   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3772     \titleemph{
3773       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3774         \spf@proof@kw
3775       }{
3776         \l__stex_sproof_spf_type_tl
3777       }
3778     }:
3779   }
3780   {~#2}
3781   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3782   \def\pst@label{}
3783   \newcount\pst@count% initialize the labeling mechanism
3784   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3785   }{
3786     \end{pst@with@label}\end{description}
3787   }
3788   \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3789   \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3790 \newcommand\spfidea[2][]{
3791   \__stex_sproof_spf_args:n{#1}
3792   \titleemph{
3793     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3794       \l__stex_sproof_spf_type_tl
3795     }:
3796   }~#2
3797   \sproofend
3798 }

```

(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

```

3799 \newenvironment{spfstep}[1][]{
3800   \_stex_sproof_spf_args:n{#1}
3801   \@in@omtexttrue
3802   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3803     \item[\the@pst@label]
3804   }
3805   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3806     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3807   }
3808   %\sref@label@id{\pst@label}
3809   \ignorespacesandpars
3810 }{
3811   \next@pst@label\ignorespacesandpars
3812 }

```

sproofcomment

```

3813 \newenvironment{sproofcomment}[1][]{
3814   \_stex_sproof_spf_args:n{#1}
3815   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3816     \item[\the@pst@label]
3817   }
3818 }{
3819   \next@pst@label
3820 }

```

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.

```

3821 \newenvironment{subproof}[2][]{
3822   \_stex_sproof_spf_args:n{#1}
3823   \def\@test{#2}
3824   \ifx\@test\empty\else
3825     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3826       \item[\the@pst@label]
3827     }{#2}
3828   \fi
3829   \begin{pst@with@label}{\pst@label,\number\count_ten}
3830 }{
3831   \end{pst@with@label}\next@pst@label
3832 }

```

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

```

3833 \newenvironment{spfcases}[2][]{
3834   \def\@test{#1}
3835   \ifx\@test\empty
3836     \begin{subproof}[method=by-cases]{#2}
3837   \else
3838     \begin{subproof}[#1,method=by-cases]{#2}
3839   \fi
3840 }{

```

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

```

3841 \end{subproof}
3842 }

```

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

```

3843 \newenvironment{spfcase}[2] [] {
3844   \__stex_sproof_spf_args:n{#1}
3845   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3846     \item[\the@pst@label]
3847   }
3848   \def\@test{#2}
3849   \ifx\@test\@empty
3850   \else
3851     {\titleemph{#2}:~}
3852   \fi
3853   \begin{pst@with@label}{\pst@label,\number\count_ten}
3854   }{
3855     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3856       \sproofend
3857     }
3858     \end{pst@with@label}
3859     \next@pst@label
3860   }

```

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

```

3861 \newcommand\spfcasesketch[3] [] {
3862   \__stex_sproof_spf_args:n{#1}
3863   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3864     \item[\the@pst@label]
3865   }
3866   \def\@test{#2}
3867   \ifx\@test\@empty
3868   \else
3869     {\titleemph{#2}:~}
3870   \fi#3
3871   \next@pst@label
3872 }%

```

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.

```

3873 \keys_define:nn { stex / just }{
3874   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3875   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3876   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3877   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3878 }

```

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

```
3879 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3880 \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.

```
3881 \newcommand\justarg[2] [] {#2}
```

```
3882 \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

```
3883 <*package>
3884
3885 %%%%%%%%%% others.dtx %%%%%%%%%%
3886
3887 <@@=stex_others>
    Warnings and error messages
3888 % None

\MSC Math subject classifier

3889 \NewDocumentCommand \MSC {m} {
3890 % TODO
3891 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3892 \@ifpackageloaded{tikzinput}{
3893 \RequirePackage{stex-tikzinput}
3894 }{}
3895 </package>
```

Chapter 29

STEX -Metatheory Implementation

```
3896 <*package>
3897 <@@=stex_modules>
3898
3899 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3900
3901 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3902 \begingroup
3903 \stex_module_setup:nn{
3904   ns=\c_stex_metatheory_ns_str,
3905   meta=NONE
3906 }{Metatheory}
3907 \stex_reactivate_macro:N \symdecl
3908 \stex_reactivate_macro:N \notation
3909 \stex_reactivate_macro:N \symdef
3910 \ExplSyntaxOff
3911 \csname stex_suppress_html:n\endcsname{
3912   % is-a (a:A, a \in A, a is an A, etc.)
3913   \symdecl[args=ai]{isa}
3914   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3915   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3916   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3917
3918   % bind (\forall, \Pi, \lambda etc.)
3919   \symdecl[args=Bi]{bind}
3920   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3921   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3922   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\;)}{#1 \comp, #2}
3923
3924   % dummy variable
3925   \symdecl{dummyvar}
3926   \notation[underscore]{dummyvar}{\comp\_}
3927   \notation[dot]{dummyvar}{\comp\cdot}
3928   \notation[dash]{dummyvar}{\comp{\rm --}}
3929
3930   %fromto (function space, Hom-set, implication etc.)
```



```

3931 \symdecl[args=ai]{fromto}
3932 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3933 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3934
3935 % mapto (lambda etc.)
3936 %\symdecl[args=Bi]{mapto}
3937 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3938 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3939 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3940
3941 % function/operator application
3942 \symdecl[args=ia]{apply}
3943 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3944 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3945
3946 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3947 \symdecl{collection}
3948 \notation[U]{collection}{\comp{\mathcal{U}}}
3949 \notation[set]{collection}{\comp{\textsf{Set}}}
3950
3951 % sequences
3952 \symdecl[args=1]{seqtype}
3953 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3954
3955 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
3956 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
3957
3958 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses\,}#1_{#3}}
3959 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses\,}#1^{#3}}
3960 % ^ superceded by \aseqfromto and \livar/\uivar
3961
3962 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses\,}}{#1\comp,#2}
3963 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\,}#2}{#1\comp,#2}
3964 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\,}#2\comp{\,\ellipses\,}#3}{#1\comp,#2}
3965
3966 % letin (‘‘let’’, local definitions, variable substitution)
3967 \symdecl[args=bii]{letin}
3968 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3969 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3970 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3971
3972 % structures
3973 \symdecl*[args=1]{module-type}
3974 \notation{module-type}{\mathtt{MOD} #1}
3975 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3976 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3977
3978 }
3979 \ExplSyntaxOn
3980 \stex_add_to_current_module:n{
3981   \let\nappa\apply
3982   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3983   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
3984   \def\livar{\csname sequence-index\endcsname[li]}

```

```

3985 \def\uivar{\csname sequence-index\endcsname[ui]}
3986 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3987 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3988 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3989 }
3990 \__stex_modules_end_module:
3991 \endgroup
3992 \</package>

```

Chapter 30

Tikzinput Implementation

```
3993 <*package>
3994
3995 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3996
3997 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3998 \RequirePackage{l3keys2e}
3999
4000 \keys_define:nn { tikzinput } {
4001   image .bool_set:N = \c_tikzinput_image_bool,
4002   image .default:n = false ,
4003   unknown .code:n = {}
4004 }
4005
4006 \ProcessKeysOptions { tikzinput }
4007
4008 \bool_if:NTF \c_tikzinput_image_bool {
4009   \RequirePackage{graphicx}
4010
4011   \providecommand\usetikzlibrary[]{}
4012   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4013 }{
4014   \RequirePackage{tikz}
4015   \RequirePackage{standalone}
4016
4017   \newcommand \tikzinput [2] [] {
4018     \setkeys{Gin}{#1}
4019     \ifx \Gin@ewidth \Gin@exclamation
4020       \ifx \Gin@eheight \Gin@exclamation
4021         \input { #2 }
4022       \else
4023         \resizebox{!}{ \Gin@eheight }{
4024           \input { #2 }
4025         }
4026       \fi
4027     \else
4028       \ifx \Gin@eheight \Gin@exclamation
4029         \resizebox{ \Gin@ewidth }{!}{
4030           \input { #2 }
4031         }
4032       \else
4033         \input { #2 }
4034       \fi
4035     \fi
4036   }
4037 }
```

```

4031     }
4032     \else
4033         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4034             \input { #2 }
4035         }
4036     \fi
4037 \fi
4038 }
4039 }
4040
4041 \newcommand \ctikzinput [2] [] {
4042     \begin{center}
4043         \tikzinput [1] {#2}
4044     \end{center}
4045 }
4046
4047 \@ifpackageloaded{stex}{
4048     \RequirePackage{stex-tikzinput}
4049 }{}
4050
4051 </package>
4052 <*stex>
4053 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4054 \RequirePackage{stex}
4055 \RequirePackage{tikzinput}
4056
4057 \newcommand\mhtikzinput [2] [] {%
4058     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4059     \stex_in_repository:nn\Gin@mhrepos{
4060         \tikzinput [1]{\mhpath{##1}{#2}}
4061     }
4062 }
4063 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4064 </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.

```
4065 \*cls)
4066 \@@=document_structure)
4067 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4068 \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`

```
4069 \keys_define:nn{ document-structure / pkg }{
4070   class      .str_set_x:N = \c_document_structure_class_str,
4071   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4072   report     .code:n      = {
4073     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4074     \str_set:Nn \c_document_structure_class_str {report}
4075   },
4076   book       .code:n      = {
4077     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4078     \str_set:Nn \c_document_structure_class_str {book}
4079   },
4080   bookpart   .code:n      = {
4081     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4082     \str_set:Nn \c_document_structure_class_str {book}
4083     \str_set:Nn \c_document_structure_topsect_str {chapter}
4084   },
```

```

4085 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4086 unknown     .code:n      = {
4087   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4088 }
4089 }
4090 \ProcessKeysOptions{ document-structure / pkg }
4091 \str_if_empty:NT \c_document_structure_class_str {
4092   \str_set:Nn \c_document_structure_class_str {article}
4093 }
4094 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4095   {\c_document_structure_class_str}
4096

```

31.3 Beefing up the document environment

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

```

4097 \RequirePackage{omdoc}
4098 \bool_if:NF \c_document_structure_minimal_bool {
4099   \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

```

4100 \keys_define:nn { document-structure / document }{
4101   id .str_set_x:N = \c_document_structure_document_id_str
4102 }
4103 \let\__document_structure_orig_document=\document
4104 \renewcommand{\document}[1][]{
4105   \keys_set:nn{ document-structure / document }{ #1 }
4106   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4107   \__document_structure_orig_document
4108 }

```

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

```

4109 }
4110 \</cls>

```

31.4 Implementation: OMDoc Package

```

4111 \<*package>
4112 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4113 \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

```

4114
4115 \keys_define:nn{ document-structure / pkg }{
4116   class      .str_set_x:N = \c_document_structure_class_str,
4117   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4118   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4119 }
4120 \ProcessKeysOptions{ document-structure / pkg }
4121 \str_if_empty:NT \c_document_structure_class_str {
4122   \str_set:Nn \c_document_structure_class_str {article}
4123 }
4124 \str_if_empty:NT \c_document_structure_topsect_str {
4125   \str_set:Nn \c_document_structure_topsect_str {section}
4126 }

```

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

```

4127 \RequirePackage{xspace}
4128 \RequirePackage{comment}
4129 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4130 \@ifpackageloaded{babel}{
4131   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4132   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4133     \input{omdoc-ngerman.ldf}
4134   }
4135 }{}
4136 %\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.

```

4137 \int_new:N \l_document_structure_section_level_int
4138 \str_case:VnF \c_document_structure_topsect_str {
4139   {part}{
4140     \int_set:Nn \l_document_structure_section_level_int {0}
4141   }
4142   {chapter}{
4143     \int_set:Nn \l_document_structure_section_level_int {1}
4144   }
4145 }{
4146   \str_case:VnF \c_document_structure_class_str {
4147     {book}{
4148       \int_set:Nn \l_document_structure_section_level_int {0}
4149     }
4150     {report}{
4151       \int_set:Nn \l_document_structure_section_level_int {0}
4152     }
4153   }{
4154     \int_set:Nn \l_document_structure_section_level_int {2}
4155   }
4156 }

```

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

```
4157 \def\current@section@level{document}%
4158 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4159 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4160 \cs_new_protected:Npn \skipomgroup {
4161   \ifcase\l_document_structure_section_level_int
4162   \or\stepcounter{part}
4163   \or\stepcounter{chapter}
4164   \or\stepcounter{section}
4165   \or\stepcounter{subsection}
4166   \or\stepcounter{subsubsection}
4167   \or\stepcounter{paragraph}
4168   \or\stepcounter{subparagraph}
4169   \fi
4170 }
```

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

`blindomgroup`

```
4171 \newcommand\at@begin@blindomgroup[1]{%
4172 \newenvironment{blindomgroup}
4173 {
4174   \int_incr:N\l_document_structure_section_level_int
4175   \at@begin@blindomgroup\l_document_structure_section_level_int
4176 }{}}
```

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

```
4177 \newcommand\omgroup@nonum[2]{%
4178   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4179   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4180 }
```

(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.

```
4181 \newcommand\omgroup@num[2]{%
```

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


```

4182 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4183   \@nameuse{#1}{#2}
4184 }{
4185   \cs_if_exist:NTF\rdfmata@sectioning{
4186     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4187   }{
4188     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4189   }
4190 }
4191 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4192 }

```

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

omgroup

```

4193 \keys_define:nn { document-structure / omgroup }{
4194   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4195   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4196   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4197   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4198   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4199   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4200   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4201   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4202   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4203   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4204 }
4205 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
4206   \str_clear:N \l__document_structure_omgroup_id_str
4207   \str_clear:N \l__document_structure_omgroup_date_str
4208   \clist_clear:N \l__document_structure_omgroup_creators_clist
4209   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4210   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4211   \tl_clear:N \l__document_structure_omgroup_type_tl
4212   \tl_clear:N \l__document_structure_omgroup_short_tl
4213   \tl_clear:N \l__document_structure_omgroup_display_tl
4214   \tl_clear:N \l__document_structure_omgroup_intro_tl
4215   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4216   \keys_set:nn { document-structure / omgroup } { #1 }
4217 }

```

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.

```

4218 \newif\if@mainmatter\@mainmattertrue
4219 \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.

```

4220 \keys_define:nn { document-structure / sectioning }{
4221   name .str_set_x:N = \l__document_structure_sect_name_str ,
4222   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4223   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4224   num .bool_set:N = \l__document_structure_sect_num_bool ,
4225 }

```

```

4226 \cs_new_protected:Nn \__document_structure_sect_args:n {
4227   \str_clear:N \l__document_structure_sect_name_str
4228   \str_clear:N \l__document_structure_sect_ref_str
4229   \bool_set_false:N \l__document_structure_sect_clear_bool
4230   \bool_set_false:N \l__document_structure_sect_num_bool
4231   \keys_set:nn { document-structure / sectioning } { #1 }
4232 }
4233 \newcommand\omdoc@sectioning[3][]{
4234   \__document_structure_sect_args:n {#1}
4235   \let\omdoc@sect@name\l__document_structure_sect_name_str
4236   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4237   \if@mainmatter% numbering not overridden by frontmatter, etc.
4238     \bool_if:NTF \l__document_structure_sect_num_bool {
4239       \omgroup@num{#2}{#3}
4240     }{
4241       \omgroup@nonum{#2}{#3}
4242     }
4243     \def\current@section@level{\omdoc@sect@name}
4244   \else
4245     \omgroup@nonum{#2}{#3}
4246   \fi
4247 }% 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.

```

4248 \newcommand\omgroup@redefine@addtocontents[1]{%
4249   %\edef\__document_structureimport{#1}%
4250   %\@for\@I:=\__document_structureimport\do{%
4251     %\edef\@path{\csname module@\@I @path\endcsname}%
4252     %\@ifundefined{tf@toc}\relax%
4253     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4254   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4255   %\def\addcontentsline##1##2##3{%
4256     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4257   %\else% hyperref.sty not loaded
4258   %\def\addcontentsline##1##2##3{%
4259     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4260   %\fi
4261 }% 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.

```

4262 \int_new:N \l_document_structure_omgroup_level_int
4263 \newenvironment{omgroup}[2][]{% keys, title
4264 {
4265   \__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.

```

4266 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4267   \omgroup@redefine@addtocontents{
4268     %\@ifundefined{module@id}\used@modules%
4269     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4270     }
4271 }

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

4272 \int_incr:N \l_document_structure_omgroup_level_int
4273 \int_incr:N \l_document_structure_section_level_int
4274 \ifcase\l_document_structure_section_level_int
4275   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4276   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4277   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4278   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4279   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4280   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4281   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4282 \fi
4283 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4284 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4285 }% for customization
4286 {}

```

and finally, we localize the sections

```

4287 \newcommand\omdoc@part@kw{Part}
4288 \newcommand\omdoc@chapter@kw{Chapter}
4289 \newcommand\omdoc@section@kw{Section}
4290 \newcommand\omdoc@subsection@kw{Subsection}
4291 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4292 \newcommand\omdoc@paragraph@kw{paragraph}
4293 \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`

```

4294 \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).

```

4295 \cs_if_exist:NTF\frontmatter{
4296   \let\__document_structure_orig_frontmatter\frontmatter
4297   \let\frontmatter\relax
4298 }{
4299   \tl_set:Nn\__document_structure_orig_frontmatter{
4300     \clearpage
4301     \@mainmatterfalse
4302     \pagenumbering{roman}
4303   }
4304 }
4305 \cs_if_exist:NTF\backmatter{

```

```

4306 \let\__document_structure_orig_backmatter\backmatter
4307 \let\backmatter\relax
4308 }{
4309 \tl_set:Nn\__document_structure_orig_backmatter{
4310 \clearpage
4311 \@mainmatterfalse
4312 \pagenumbering{roman}
4313 }
4314 }

```

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.

```

4315 \newenvironment{frontmatter}{
4316 \__document_structure_orig_frontmatter
4317 }{
4318 \cs_if_exist:NTF\mainmatter{
4319 \mainmatter
4320 }{
4321 \clearpage
4322 \@mainmattertrue
4323 \pagenumbering{arabic}
4324 }
4325 }

```

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

```

4326 \newenvironment{backmatter}{
4327 \__document_structure_orig_backmatter
4328 }{
4329 \cs_if_exist:NTF\mainmatter{
4330 \mainmatter
4331 }{
4332 \clearpage
4333 \@mainmattertrue
4334 \pagenumbering{arabic}
4335 }
4336 }

```

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

```

4337 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4338 \newcommand\afterprematurestop{}
4339 \def\prematurestop@endomgroup{
4340 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4341 \end{omgroup}
4342 \prematurestop@endomgroup
4343 }
4344 }
4345 \providecommand\prematurestop{
4346 \message{Stopping~sTeX~processing~prematurely}

```

```

4347 \prematuarestop@endomgroup
4348 \afterprematuarestop
4349 \end{document}
4350 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

4351 \RequirePackage{etoolbox}
4352 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4353 \newrobustcmd\useSGvar[1]{%
4354 \@ifundefined{sTeX@Gvar@#1}
4355 {\PackageError{omdoc}
4356 {The sTeX Global variable #1 is undefined}
4357 {set it with \protect\setSGvar}}
4358 \@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.

```

4359 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4360 \@ifundefined{sTeX@Gvar@#1}
4361 {\PackageError{omdoc}
4362 {The sTeX Global variable #1 is undefined}
4363 {set it with \protect\setSGvar}}
4364 {\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.

```
4365 \*cls)
4366 \@@=mikoslides)
4367 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4368 \RequirePackage{l3keys2e,expl-keystr-compat}
4369
4370 \keys_define:nn{mikoslides / cls}{
4371   class .code:n = {
4372     \PassOptionsToClass{\CurrentOption}{omdoc}
4373     \str_if_eq:nnT{#1}{book}{
4374       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4375     }
4376     \str_if_eq:nnT{#1}{report}{
4377       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4378     }
4379   },
4380   notes .bool_set:N = \c__mikoslides_notes_bool ,
4381   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4382   unknown .code:n = {
4383     \PassOptionsToClass{\CurrentOption}{omdoc}
4384     \PassOptionsToClass{\CurrentOption}{beamer}
4385     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4386   }
4387 }
4388 \ProcessKeysOptions{ mikoslides / cls }
4389 \bool_if:NTF \c__mikoslides_notes_bool {
4390   \PassOptionsToPackage{notes=true}{mikoslides}
4391 }{
4392   \PassOptionsToPackage{notes=false}{mikoslides}
4393 }
4394 \</cls)
```

now we do the same for the mikoslides package.

```

4395 \*package>
4396 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4397 \RequirePackage{l3keys2e,expl-keystr-compat}
4398
4399 \keys_define:nn{mikoslides / pkg}{
4400   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
4401   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4402   notes            .bool_set:N = \c__mikoslides_notes_bool ,
4403   slides            .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4404   sectocframes      .bool_set:N = \c__mikoslides_sectocframes_bool ,
4405   frameimages       .bool_set:N = \c__mikoslides_frameimages_bool ,
4406   fiboxed           .bool_set:N = \c__mikoslides_fiboxed_bool ,
4407   noproblems        .bool_set:N = \c__mikoslides_noproblems_bool,
4408   unknown           .code:n      = {
4409     \PassOptionsToClass{\CurrentOption}{stex}
4410     \PassOptionsToClass{\CurrentOption}{tikzinput}
4411   }
4412 }
4413 \ProcessKeysOptions{ mikoslides / pkg }
4414 \newif\ifnotes
4415 \bool_if:NTF \c__mikoslides_notes_bool {
4416   \notesttrue
4417 }{
4418   \notesfalse
4419 }
4420

```

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

```

4421 \str_if_empty:NTF \c__mikoslides_topsect_str {
4422   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4423 }{
4424   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4425 }
4426 \</package>

```

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

```

4427 \*cls>
4428 \bool_if:NTF \c__mikoslides_notes_bool {
4429   \LoadClass{omdoc}
4430 }{
4431   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4432   \newcounter{Item}
4433   \newcounter{paragraph}
4434   \newcounter{subparagraph}
4435   \newcounter{Hfootnote}
4436   \RequirePackage{omdoc}
4437 }

```

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

```

4438 \RequirePackage{mikoslides}
4439 \</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.

```

4440 \*package>
4441 \bool_if:NT \c__mikoslides_notes_bool {
4442   \RequirePackage{a4wide}
4443   \RequirePackage{marginnote}
4444   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4445   \RequirePackage{mdframed}
4446   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4447   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4448 }
4449 \RequirePackage{stex-compatibility}
4450 \RequirePackage{stex-tikzinput}
4451 \RequirePackage{etoolbox}
4452 \RequirePackage{amssymb}
4453 \RequirePackage{amsmath}
4454 \RequirePackage{comment}
4455 \RequirePackage{textcomp}
4456 \RequirePackage{url}
4457 \RequirePackage{graphicx}
4458 \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

```

4459 \bool_if:NT \c__mikoslides_notes_bool {
4460   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4461 }

```

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

```

4462 \newcounter{slide}
4463 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4464 \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.

```

4465 \bool_if:NTF \c__mikoslides_notes_bool {
4466   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4467 }{
4468   \excludecomment{note}
4469 }

```

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.

```
4470 \bool_if:NT \c__mikoslides_notes_bool {
4471   \newlength{\slideframewidth}
4472   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4473 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4474   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4475     \bool_set_true:N #1
4476   }{
4477     \bool_set_false:N #1
4478   }
4479 }
4480 \keys_define:nn{mikoslides / frame}{
4481   label .str_set_x:N = \l__mikoslides_frame_label_str,
4482   allowframebreaks .code:n = {
4483     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4484   },
4485   allowdisplaybreaks .code:n = {
4486     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4487   },
4488   fragile .code:n = {
4489     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4490   },
4491   shrink .code:n = {
4492     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4493   },
4494   squeeze .code:n = {
4495     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4496   },
4497   t .code:n = {
4498     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4499   },
4500 }
4501 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4502   \str_clear:N \l__mikoslides_frame_label_str
4503   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4504   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4505   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4506   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4507   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4508   \bool_set_true:N \l__mikoslides_frame_t_bool
4509   \keys_set:nn { mikoslides / frame }{ #1 }
4510 }
```

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

```
4511 \renewenvironment{frame}[1][]{
4512   \__mikoslides_frame_args:n{#1}
4513   \sffamily
4514   \stepcounter{slide}
4515   \def\@currentlabel{\theslide}
4516   \str_if_empty:NF \l__mikoslides_frame_label_str {
4517     \label{\l__mikoslides_frame_label_str}
```

```
4518 }
```

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

```
4519 \def\itemize@level{outer}
4520 \def\itemize@outer{outer}
4521 \def\itemize@inner{inner}
4522 \renewcommand\newpage{\addtocounter{framenum}{1}}
4523 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4524 \renewenvironment{itemize}{
4525   \ifx\itemize@level\itemize@outer
4526     \def\itemize@label{$\rhd$}
4527   \fi
4528   \ifx\itemize@level\itemize@inner
4529     \def\itemize@label{$\scriptstyle\rhd$}
4530   \fi
4531   \begin{list}
4532   {\itemize@label}
4533   {\setlength{\labelsep}{.3em}
4534    \setlength{\labelwidth}{.5em}
4535    \setlength{\leftmargin}{1.5em}
4536   }
4537   \edef\itemize@level{\itemize@inner}
4538 }{
4539   \end{list}
4540 }
```

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

```
4541 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4542 }{
4543   \medskip\miko@slidelabel\end{mdframed}
4544 }
```

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

`\frametitle`

```
4545 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4546 }
```

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

EdN:18

`\pause` 18

```
4547 \bool_if:NT \c__mikoslides_notes_bool {
4548   \newcommand\pause{}
4549 }
```

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

`nomtext`

```
4550 \bool_if:NTF \c__mikoslides_notes_bool {
4551   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4552 }{
4553   \excludecomment{nomtext}
4554 }
```

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

```

nomgroup
4555 \bool_if:NTF \c__mikoslides_notes_bool {
4556   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4557 }{
4558   \excludecomment{nomgroup}
4559 }

ndefinition
4560 \bool_if:NTF \c__mikoslides_notes_bool {
4561   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4562 }{
4563   \excludecomment{ndefinition}
4564 }

nassertion
4565 \bool_if:NTF \c__mikoslides_notes_bool {
4566   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4567 }{
4568   \excludecomment{nassertion}
4569 }

nsproof
4570 \bool_if:NTF \c__mikoslides_notes_bool {
4571   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4572 }{
4573   \excludecomment{nsproof}
4574 }

nexample
4575 \bool_if:NTF \c__mikoslides_notes_bool {
4576   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4577 }{
4578   \excludecomment{nexample}
4579 }

\inputref@*skip We customize the hooks for in \inputref.
4580 \def\inputref@preskip{\smallskip}
4581 \def\inputref@postskip{\medskip}

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

\inputref*
4582 \let\orig@inputref\inputref
4583 \def\inputref{\@ifstar\ninputref\orig@inputref}
4584 \newcommand\ninputref[2] [] {
4585   \bool_if:NT \c__mikoslides_notes_bool {
4586     \orig@inputref[#1]{#2}
4587   }
4588 }

(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>}`.

```

4589 \newlength{\slidelogoheight}
4590
4591 \bool_if:NTF \c__mikoslides_notes_bool {
4592   \setlength{\slidelogoheight}{.4cm}
4593 }{
4594   \setlength{\slidelogoheight}{1cm}
4595 }
4596 \newsavebox{\slidelogo}
4597 \sbox{\slidelogo}{\text{\TeX}}
4598 \newrobustcmd{\setslidelogo}[1]{
4599   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4600 }
```

(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.

```

4601 \def\source{Michael Kohlhase}% customize locally
4602 \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.

```

4603 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4604 \newsavebox{\cclogo}
4605 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4606 \newif\ifcchref\cchreffalse
4607 \AtBeginDocument{
4608   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4609 }
4610 \def\licensing{
4611   \ifcchref
4612     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4613   \else
4614     {\usebox{\cclogo}}
4615   \fi
4616 }
4617 \newrobustcmd{\setlicensing}[2][]{
4618   \def\@url{#1}
4619   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4620   \ifx\@url\@empty
4621     \def\licensing{{\usebox{\cclogo}}}
4622   \else
4623     \def\licensing{
```

```

4624     \ifcchref
4625     \href{#1}{\usebox{\cclogo}}
4626   \else
4627     {\usebox{\cclogo}}
4628   \fi
4629 }
4630 \fi
4631 }

```

(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

```

4632 \newrobustcmd\miko@slidelabel{
4633   \vbox to \slidelogoheight{
4634     \vss\hbox to \slidewidth
4635     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4636   }
4637 }

```

(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.

```

4638 \def\Gin@mhrepos{}
4639 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4640 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
4641 \newrobustcmd\frameimage[2][{}{
4642   \stepcounter{slide}
4643   \bool_if:NT \c__mikoslides_frameimages_bool {
4644     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4645     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4646     \begin{center}
4647       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4648         \fbox{
4649           \ifx\Gin@ewidth\@empty
4650             \ifx\Gin@mhrepos\@empty
4651               \mhgraphics[width=\slidewidth,#1]{#2}
4652             \else
4653               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4654             \fi
4655           \else% \Gin@ewidth empty
4656             \ifx\Gin@mhrepos\@empty
4657               \mhgraphics[#1]{#2}
4658             \else
4659               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4660             \fi
4661           \fi% \Gin@ewidth empty
4662         }
4663       }{
4664         \ifx\Gin@ewidth\@empty

```

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

```

4665         \ifx\Gin@mhrepos\empty
4666         \mhgraphics[width=\slidewidth,#1]{#2}
4667     \else
4668         \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4669     \fi
4670     \ifx\Gin@mhrepos\empty
4671         \mhgraphics[#1]{#2}
4672     \else
4673         \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4674     \fi
4675     \fi% Gin@ewidth empty
4676 }
4677 \end{center}
4678 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4679 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4680 }
4681 } % 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.

```

4682 \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

```

4683 \AddToHook{begindocument}{
4684     \definecolor{green}{rgb}{0,.5,0}
4685     \definecolor{purple}{cmyk}{.3,1,0,.17}
4686 }

```

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`.

```

4687 % \def\STpresent#1{\textcolor{blue}{#1}}
4688 \def\defemph#1{\textcolor{magenta}{#1}}
4689 \def\symrefemph#1{\textcolor{cyan}{#1}}
4690 \def\compemph#1{\textcolor{blue}{#1}}
4691 \def\titleemph#1{\textcolor{blue}{#1}}
4692 \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.

```

4693 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4694 \def\smalltextwarning{
4695     \pgfuseimage{miko@small@dbend}
4696     \xspace
4697 }
4698 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4699 \newrobustcmd\textwarning{
4700   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4701   \xspace
4702 }
4703 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4704 \newrobustcmd\bigtextwarning{
4705   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4706   \xspace
4707 }

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

4708 \newrobustcmd\putgraphicsat[3]{
4709   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4710 }
4711 \newrobustcmd\putat[2]{
4712   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4713 }

```

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`.

```

4714 \bool_if:NT \c__mikoslides_sectocframes_bool {
4715   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4716     \newcounter{chapter}\counterwithin*{section}{chapter}
4717   }{
4718     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4719       \newcounter{chapter}\counterwithin*{section}{chapter}
4720     }
4721   }
4722 }

```

`\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

4723 \def\part@prefix{}
4724 \@ifpackageloaded{omdoc}{}{
4725   \str_case:VnF \__mikoslidestopsect {
4726     {part}{
4727       \int_set:Nn \l_document_structure_section_level_int {0}
4728       \def\thesection{\arabic{chapter}.\arabic{section}}
4729       \def\part@prefix{\arabic{chapter}.}
4730     }
4731     {chapter}{
4732       \int_set:Nn \l_document_structure_section_level_int {1}
4733       \def\thesection{\arabic{chapter}.\arabic{section}}
4734       \def\part@prefix{\arabic{chapter}.}
4735     }
4736   }{
4737     \int_set:Nn \l_document_structure_section_level_int {2}
4738     \def\part@prefix{}

```

```

4739 }
4740 }
4741
4742 \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

```

4743 \renewenvironment{omgroup}[2][]{
4744   \__document_structure_omgroup_args:n { #1 }
4745   \int_incr:N \l_document_structure_omgroup_level_int
4746   \int_incr:N \l_document_structure_section_level_int
4747   \bool_if:NT \c__mikoslides_sectocframes_bool {
4748     \stepcounter{slide}
4749     \begin{frame}[noframenumbering]
4750     \vfill\Large\centering
4751     \red{
4752       \ifcase\l_document_structure_section_level_int\or
4753         \stepcounter{part}
4754         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4755         \def\currentsectionlevel{\omdoc@part@kw}
4756       \or
4757         \stepcounter{chapter}
4758         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4759         \def\currentsectionlevel{\omdoc@chapter@kw}
4760       \or
4761         \stepcounter{section}
4762         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4763         \def\currentsectionlevel{\omdoc@section@kw}
4764       \or
4765         \stepcounter{subsection}
4766         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4767         \def\currentsectionlevel{\omdoc@subsection@kw}
4768       \or
4769         \stepcounter{subsubsection}
4770         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4771         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4772       \or
4773         \stepcounter{paragraph}
4774         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
4775         \def\currentsectionlevel{\omdoc@paragraph@kw}
4776       \else
4777         \def\__mikoslideslabel{}
4778         \def\currentsectionlevel{\omdoc@paragraph@kw}
4779       \fi% end ifcase
4780       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4781       \quad #2%
4782     }%
4783     \vfill%
4784     \end{frame}%
4785   }
4786   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```



```

4787 }{}
4788 }

```

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

```

4789 \def\inserttheorembodyfont{\normalfont}
4790 \bool_if:NF \c__mikoslides_notes_bool {
4791   \defbeamertemplate{theorem begin}{miko}
4792   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4793     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4794     \inserttheorempunctuation\inserttheorembodyfont\space}
4795   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

4796   \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.

```

4797   \expandafter\def\csname Parent2\endcsname{}
4798 }
4799 \bool_if:NT \c__mikoslides_notes_bool {
4800   \renewenvironment{columns}[1][]{%
4801     \par\noindent%
4802     \begin{minipage}%
4803       \slidewidth\centering\leavevmode%
4804     }{%
4805       \end{minipage}\par\noindent%
4806     }%
4807   \newsavebox\columnbox%
4808   \renewenvironment<>{column}[2][]{%
4809     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4810     }{%
4811       \end{minipage}\end{lrbox}\usebox\columnbox%
4812     }%
4813 }
4814 \bool_if:NTF \c__mikoslides_noproblems_bool {
4815   \newenvironment{problems}{}{}
4816 }{
4817   \excludecomment{problems}
4818 }

```

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.

```

4819 \gdef\printexcursions{}
4820 \newcommand\excursionref[2]{% label, text
4821   \bool_if:NT \c__mikoslides_notes_bool {
4822     \begin{sparagraph}[title=Excursion]
4823       #2 \sref[fallback=the appendix]{#1}.
4824     \end{sparagraph}
4825   }

```

```

4826 }
4827 \newcommand\activate@excursion[2][]{
4828   \gappto\printexcursions{\inputref{#1}{#2}}
4829 }
4830 \newcommand\excursion[4][]{% repos, label, path, text
4831   \bool_if:NT \c__mikoslides_notes_bool {
4832     \activate@excursion[1]{#3}\excursionref{#2}{#4}
4833   }
4834 }

```

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

\excursiongroup

```

4835 \keys_define:nn{mikoslides / excursiongroup }{
4836   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4837   intro       .tl_set:N    = \l__mikoslides_excursion_intro_tl,
4838   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4839 }
4840 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4841   \tl_clear:N \l__mikoslides_excursion_intro_tl
4842   \str_clear:N \l__mikoslides_excursion_id_str
4843   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4844   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4845 }
4846 \newcommand\excursiongroup[1][]{
4847   \__mikoslides_excursion_args:n{ #1 }
4848   \ifdefempty\printexcursions{}% only if there are excursions
4849   {\begin{note}
4850     \begin{omgroup}[#1]{Excursions}%
4851     \ifdefempty\l__mikoslides_excursion_intro_tl{\{
4852       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4853         \l__mikoslides_excursion_intro_tl
4854       }
4855     }
4856     \printexcursions%
4857     \end{omgroup}
4858   \end{note}}
4859 }
4860 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
4861 \</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.

```
4862 <*package>
4863 <@@=problems>
4864 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4865 \RequirePackage{l3keys2e,expl-keystr-compatible}
4866
4867 \keys_define:nn { problem / pkg }{
4868   notes      .default:n    = { true },
4869   notes      .bool_set:N   = \c__problems_notes_bool,
4870   gnotes     .default:n    = { true },
4871   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4872   hints      .default:n    = { true },
4873   hints      .bool_set:N   = \c__problems_hints_bool,
4874   solutions  .default:n    = { true },
4875   solutions  .bool_set:N   = \c__problems_solutions_bool,
4876   pts        .default:n    = { true },
4877   pts        .bool_set:N   = \c__problems_pts_bool,
4878   min        .default:n    = { true },
4879   min        .bool_set:N   = \c__problems_min_bool,
4880   boxed      .default:n    = { true },
4881   boxed      .bool_set:N   = \c__problems_boxed_bool,
4882   unknown    .code:n       = {}
4883 }
4884 \def\solutionstrue{
4885   \bool_set_true:N \c__problems_solutions_bool
4886 }
4887 \def\solutionsfalse{
4888   \bool_set_false:N \c__problems_solutions_bool
4889 }
4890
4891 \ProcessKeysOptions{ problem / pkg }
```

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

```

4892 \RequirePackage{stex-compatibility}
4893 \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.

```

4894 \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.

```

4895 \def\prob@problem@kw{Problem}
4896 \def\prob@solution@kw{Solution}
4897 \def\prob@hint@kw{Hint}
4898 \def\prob@note@kw{Note}
4899 \def\prob@gnote@kw{Grading}
4900 \def\prob@pt@kw{pt}
4901 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4902 \@ifpackageloaded{babel}{
4903   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4904   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4905     \input{problem-ngerman.ldf}
4906   }
4907   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4908     \input{problem-finnish.ldf}
4909   }
4910   \clist_if_in:NnT \l_tmpa_clist {french}{
4911     \input{problem-french.ldf}
4912   }
4913   \clist_if_in:NnT \l_tmpa_clist {russian}{
4914     \input{problem-russian.ldf}
4915   }
4916 }{}

```

33.2 Problems and Solutions

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

```

4917 \keys_define:nn{ problem / problem }{
4918   id      .str_set:x:N = \l__problems_prob_id_str,
4919   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4920   min     .tl_set:N    = \l__problems_prob_min_tl,
4921   title   .tl_set:N    = \l__problems_prob_title_tl,
4922   refnum  .int_set:N   = \l__problems_prob_refnum_int
4923 }
4924 \cs_new_protected:Nn \__problems_prob_args:n {
4925   \str_clear:N \l__problems_prob_id_str
4926   \tl_clear:N \l__problems_prob_pts_tl
4927   \tl_clear:N \l__problems_prob_min_tl
4928   \tl_clear:N \l__problems_prob_title_tl

```

```

4929 \int_zero_new:N \l__problems_prob_refnum_int
4930 \keys_set:nn { problem / problem }{ #1 }
4931 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4932   \let\l__problems_inclprob_refnum_int\undefined
4933 }
4934 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4935 \newcounter{problem}
4936 \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.

```

4937 \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

```

4938 \newcommand\prob@number{
4939   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4940     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4941   }{
4942     \int_if_exist:NTF \l__problems_prob_refnum_int {
4943       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4944     }{
4945       \prob@label\theproblem
4946     }
4947   }
4948 }

```

(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.

```

4949 \newcommand\prob@title[3]{%
4950   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4951     #2 \l__problems_inclprob_title_tl #3
4952   }{
4953     \tl_if_exist:NTF \l__problems_prob_title_tl {
4954       #2 \l__problems_prob_title_tl #3
4955     }{
4956       #1
4957     }
4958   }
4959 }

```

(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.

```

4960 \def\prob@heading{
4961   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4962   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4963 }

```

(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`

```

4964 \newenvironment{problem}[1][1]{
4965   \_problems_prob_args:n{#1}%\sref@target%
4966   \@in@omtexttrue% we are in a statement (for inline definitions)
4967   \stepcounter{problem}\record@problem
4968   \def\current@section@level{\prob@problem@kw}
4969   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4970 }%
4971 {\smallskip}
4972 \bool_if:NT \c__problems_boxed_bool {
4973   \surroundwithmdframed{problem}
4974 }

```

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

```

4975 \def\record@problem{
4976   \protected@write\@auxout{}
4977   {
4978     \string\@problem{\prob@number}
4979     {
4980       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4981         \l__problems_inclprob_pts_tl
4982       }{
4983         \l__problems_prob_pts_tl
4984       }
4985     }%
4986     {
4987       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4988         \l__problems_inclprob_min_tl
4989       }{
4990         \l__problems_prob_min_tl
4991       }
4992     }
4993   }
4994 }

```

(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).

```

4995 \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.

```

4996 \keys_define:nn { problem / solution }{
4997   id          .str_set_x:N = \l__problems_solution_id_str ,
4998   for         .tl_set:N    = \l__problems_solution_for_tl ,
4999   height      .dim_set:N   = \l__problems_solution_height_dim ,
5000   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5001   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5002   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5003 }
5004 \cs_new_protected:Nn \__problems_solution_args:n {
5005   \str_clear:N \l__problems_solution_id_str
5006   \tl_clear:N \l__problems_solution_for_tl
5007   \tl_clear:N \l__problems_solution_srccite_tl
5008   \clist_clear:N \l__problems_solution_creators_clist
5009   \clist_clear:N \l__problems_solution_contributors_clist
5010   \dim_zero:N \l__problems_solution_height_dim
5011   \keys_set:nn { problem / solution }{ #1 }
5012 }

```

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

```

5013 \newcommand\@startsolution[1][ ]{
5014   \__problems_solution_args:n { #1 }
5015   \@in@omtexttrue% we are in a statement.
5016   \bool_if:NF \c__problems_boxed_bool { \hrule }
5017   \smallskip\noindent
5018   {\textbf\prob@solution@kw : \enspace}
5019   \begin{small}
5020   \def\current@section@level{\prob@solution@kw}
5021   \ignorespacesandpars
5022 }

```

\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.

```

5023 \newcommand\startsolutions{
5024   \specialcomment{solution}{\@startsolution}{
5025     \bool_if:NF \c__problems_boxed_bool {
5026       \hrule\medskip
5027     }
5028     \end{small}%
5029   }
5030   \bool_if:NT \c__problems_boxed_bool {
5031     \surroundwithmdframed{solution}
5032   }
5033 }

```

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

\stopsolutions

```

5034 \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.

```

5035 \bool_if:NTF \c__problems_solutions_bool {
5036   \startsolutions
5037 }{
5038   \stopsolutions
5039 }

```

exnote

```

5040 \bool_if:NTF \c__problems_notes_bool {
5041   \newenvironment{exnote}[1][]{
5042     \par\smallskip\hrule\smallskip
5043     \noindent\textbf{\prob@note@kw : }\small
5044   }{
5045     \smallskip\hrule
5046   }
5047 }{
5048   \excludecomment{exnote}
5049 }

```

hint

```

5050 \bool_if:NTF \c__problems_notes_bool {
5051   \newenvironment{hint}[1][]{
5052     \par\smallskip\hrule\smallskip
5053     \noindent\textbf{\prob@hint@kw :~ }\small
5054   }{
5055     \smallskip\hrule
5056   }
5057   \newenvironment{exhint}[1][]{
5058     \par\smallskip\hrule\smallskip
5059     \noindent\textbf{\prob@hint@kw :~ }\small
5060   }{
5061     \smallskip\hrule
5062   }
5063 }{
5064   \excludecomment{hint}
5065   \excludecomment{exhint}
5066 }

```

gnote

```

5067 \bool_if:NTF \c__problems_notes_bool {
5068   \newenvironment{gnote}[1][]{
5069     \par\smallskip\hrule\smallskip
5070     \noindent\textbf{\prob@gnote@kw : }\small
5071   }{
5072     \smallskip\hrule
5073   }
5074 }{
5075   \excludecomment{gnote}
5076 }

```


33.3 Multiple Choice Blocks

```

5077 \newenvironment{mcb}{
5078   \begin{enumerate}
5079 }{
5080   \end{enumerate}
5081 }

```

we define the keys for the mcc macro

```

5082 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5083   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5084     \bool_set_true:N #1
5085   }{
5086     \bool_set_false:N #1
5087   }
5088 }
5089 \keys_define:nn { problem / mcc }{
5090   id          .str_set_x:N = \l__problems_mcc_id_str ,
5091   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5092   T           .default:n   = { true } ,
5093   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5094   F           .default:n   = { true } ,
5095   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5096   Ttext       .code:n      = {
5097     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5098   } ,
5099   Ftext       .code:n      = {
5100     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5101   }
5102 }
5103 \cs_new_protected:Nn \l__problems_mcc_args:n {
5104   \str_clear:N \l__problems_mcc_id_str
5105   \tl_clear:N \l__problems_mcc_feedback_tl
5106   \bool_set_true:N \l__problems_mcc_t_bool
5107   \bool_set_true:N \l__problems_mcc_f_bool
5108   \bool_set_true:N \l__problems_mcc_Ttext_bool
5109   \bool_set_false:N \l__problems_mcc_Ftext_bool
5110   \keys_set:nn { problem / mcc }{ #1 }
5111 }

```

\mcc

```

5112 \newcommand\mcc[2][] {
5113   \l__problems_mcc_args:n{ #1 }
5114   \item #2
5115   \bool_if:NT \c__problems_solutions_bool {
5116     \
5117     \bool_if:NT \l__problems_mcc_t_bool {
5118       % TODO!
5119       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5120     }
5121     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

5122      % TODO!
5123      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5124    }
5125    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5126      !
5127    }{
5128      \l__problems_mcc_feedback_tl
5129    }
5130  }
5131 } %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.

```

5132
5133 \keys_define:nn{ problem / inclproblem }{
5134   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5135   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5136   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5137   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5138   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5139   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5140 }
5141 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5142   % \str_clear:N \l__problems_prob_id_str
5143   \tl_clear:N \l__problems_inclprob_pts_tl
5144   \tl_clear:N \l__problems_inclprob_min_tl
5145   \tl_clear:N \l__problems_inclprob_title_tl
5146   \int_zero_new:N \l__problems_inclprob_refnum_int
5147   \str_clear:N \l__problems_inclprob_mhrepos_str
5148   \keys_set:nn { problem / inclproblem }{ #1 }
5149   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5150     \let\l__problems_inclprob_pts_tl\undefined
5151   }
5152   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5153     \let\l__problems_inclprob_min_tl\undefined
5154   }
5155   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5156     \let\l__problems_inclprob_title_tl\undefined
5157   }
5158   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5159     \let\l__problems_inclprob_refnum_int\undefined
5160   }
5161 }
5162
5163 \cs_new_protected:Nn \l__problems_inclprob_clear: {
5164   % \str_clear:N \l__problems_prob_id_str
5165   \let\l__problems_inclprob_pts_tl\undefined
5166   \let\l__problems_inclprob_min_tl\undefined

```

```

5167 \let\l__problems_inclprob_title_tl\undefined
5168 \let\l__problems_inclprob_refnum_int\undefined
5169 \let\l__problems_inclprob_mhrepos_str\undefined
5170 }
5171
5172 \newcommand\includeproblem[2][]{
5173   \__problems_inclprob_args:n{ #1 }
5174   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5175     \input{#2}
5176   }{
5177     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5178       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5179     }
5180   }
5181   \__problems_inclprob_clear:
5182 }

```

(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.

```

5183 \AddToHook{enddocument}{
5184   \bool_if:NT \c__problems_pts_bool {
5185     \message{Total:~\arabic{pts}~points}
5186   }
5187   \bool_if:NT \c__problems_min_bool {
5188     \message{Total:~\arabic{min}~minutes}
5189   }
5190 }

```

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

```

5191 \def\pts#1{
5192   \bool_if:NT \c__problems_pts_bool {
5193     \marginpar{#1~\prob@pt@kw}
5194   }
5195 }
5196 \def\min#1{
5197   \bool_if:NT \c__problems_min_bool {
5198     \marginpar{#1~\prob@min@kw}
5199   }
5200 }

```

\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.

```

5201 \newcounter{pts}
5202 \def\show@pts{
5203   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5204     \bool_if:NT \c__problems_pts_bool {
5205       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5206       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5207     }
5208   }{
5209     \tl_if_exist:NT \l__problems_prob_pts_tl {
5210       \bool_if:NT \c__problems_pts_bool {
5211         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5212         \addtocounter{pts}{\l__problems_prob_pts_tl}
5213       }
5214     }
5215   }
5216 }

```

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

\show@min

```

5217 \newcounter{min}
5218 \def\show@min{
5219   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5220     \bool_if:NT \c__problems_min_bool {
5221       \marginpar{\l__problems_inclprob_pts_tl;min}
5222       \addtocounter{min}{\l__problems_inclprob_min_tl}
5223     }
5224   }{
5225     \tl_if_exist:NT \l__problems_prob_min_tl {
5226       \bool_if:NT \c__problems_min_bool {
5227         \marginpar{\l__problems_prob_min_tl;min}
5228         \addtocounter{min}{\l__problems_prob_min_tl}
5229       }
5230     }
5231   }
5232 }
5233 \</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.

```
5234 <@@=hwexam>
5235 <*cls>
5236 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5237 \RequirePackage{l3keys2e,expl-keystr-compatible}
5238 \DeclareOption*{
5239   \PassOptionsToClass{\CurrentOption}{omdoc}
5240   \PassOptionsToPackage{\CurrentOption}{stex}
5241   \PassOptionsToPackage{\CurrentOption}{hwexam}
5242   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5243 }
5244 \ProcessOptions
```

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

```
5245 \LoadClass{omdoc}
5246 \RequirePackage{stex}
5247 \RequirePackage{hwexam}
5248 \RequirePackage{tikzinput}
5249 \RequirePackage{graphicx}
5250 \RequirePackage{a4wide}
5251 \RequirePackage{amssymb}
5252 \RequirePackage{amstext}
5253 \RequirePackage{amsmath}
```

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

```

5254 \newcommand\assig@default@type{\hwexam@assignment@kw}
5255 \def\document@hwexamtype{\assig@default@type}
5256 <@@=document_structure>
5257 \keys_define:nn { document-structure / document }{
5258 id .str_set_x:N = \c_document_structure_document_id_str,
5259 hwexamtype .tl_set:N = \document@hwexamtype
5260 }
5261 <@@=hwexam>
5262 </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.

```
5263 \*package>
5264 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5265 \RequirePackage{l3keys2e,expl-keystr-compat}
5266
5267 \newif\iftest\testfalse
5268 \DeclareOption{test}{\testtrue}
5269 \newif\ifmultiple\multiplefalse
5270 \DeclareOption{multiple}{\multipletrue}
5271 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5272 \ProcessOptions
```

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

```
5273 \RequirePackage{keyval}[1997/11/10]
5274 \RequirePackage{problem}
```

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

```
5275 \newcommand\hwexam@assignment@kw{Assignment}
5276 \newcommand\hwexam@given@kw{Given}
5277 \newcommand\hwexam@due@kw{Due}
5278 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5279   space}%
5280 \newcommand\correction@probs@kw{prob.}%
5281 \newcommand\correction@pts@kw{total}%
5282 \newcommand\correction@reached@kw{reached}%
5283 \newcommand\correction@sum@kw{Sum}%
5284 \newcommand\correction@grade@kw{grade}%
5285 \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

```

5286 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5287
5288 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5289 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5290   \input{hwexam-ngerman.ldf}
5291 }
5292 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5293   \input{hwexam-finnish.ldf}
5294 }
5295 \clist_if_in:NnT \l_tmpa_clist {french}{
5296   \input{hwexam-french.ldf}
5297 }
5298 \clist_if_in:NnT \l_tmpa_clist {russian}{
5299   \input{hwexam-russian.ldf}
5300 }

```

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.

```

5301 \newcounter{assignment}
5302 \numberproblemsin{assignment}
5303 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5304 \keys_define:nn { hwexam / assignment } {
5305   id .str_set:N = \l__hwexam_assign_id_str,
5306   number .int_set:N = \l__hwexam_assign_number_int,
5307   title .tl_set:N = \l__hwexam_assign_title_tl,
5308   type .tl_set:N = \l__hwexam_assign_type_tl,
5309   given .tl_set:N = \l__hwexam_assign_given_tl,
5310   due .tl_set:N = \l__hwexam_assign_due_tl,
5311   loadmodules .code:n = {
5312     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5313   }
5314 }
5315 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5316   \str_clear:N \l__hwexam_assign_id_str
5317   \int_set:Nn \l__hwexam_assign_number_int {-1}
5318   \tl_clear:N \l__hwexam_assign_title_tl
5319   \tl_clear:N \l__hwexam_assign_type_tl
5320   \tl_clear:N \l__hwexam_assign_given_tl
5321   \tl_clear:N \l__hwexam_assign_due_tl
5322   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5323   \keys_set:nn { hwexam / assignment }{ #1 }
5324 }

```

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.

```

5325 \newcommand\given@due[2]{
5326 \bool_lazy_all:nF {
5327 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5328 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5329 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5330 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5331 }{ #1 }
5332
5333 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5334 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5335 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5336 }
5337 }{
5338 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5339 }
5340
5341 \bool_lazy_or:nnF {
5342 \bool_lazy_and_p:nn {
5343 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5344 }{
5345 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5346 }
5347 }{
5348 \bool_lazy_and_p:nn {
5349 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5350 }{
5351 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5352 }
5353 }{ ,~ }
5354
5355 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5356 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5357 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5358 }
5359 }{
5360 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5361 }
5362
5363 \bool_lazy_all:nF {
5364 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5365 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5366 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5367 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5368 }{ #2 }
5369 }

```

`\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.

```

5370 \newcommand\assignment@title[3]{

```

```

5371 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5372 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5373 #1
5374 }{
5375 #2\l__hwexam_assign_title_tl#3
5376 }
5377 }{
5378 #2\l__hwexam_inclassassign_title_tl#3
5379 }
5380 }

```

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

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

```

5381 \newcommand\assignment@number{
5382 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5383 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5384 \int_use:N \l__hwexam_assign_number_int
5385 }
5386 }{
5387 \int_use:N \l__hwexam_inclassassign_number_int
5388 }
5389 }

```

(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.

```

5390 \newenvironment{assignment}[1][ ]{
5391 \__hwexam_assignment_args:n { #1 }
5392 %\sref@target
5393 \let\__hwexamnum\l__hwexam_assign_number_int
5394 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5395 \stepcounter{assignment}
5396 }{
5397 \setcounter{assignment}{\int_use:N\__hwexamnum}
5398 }
5399 \setcounter{problem}{0}
5400 \def\current@section@level{\document@hwexamtype}
5401 %\sref@label@id{\document@hwexamtype \thesection}
5402 \begin{@assignment}
5403 }{
5404 \end{@assignment}
5405 }

```

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

```

5406 \def\__hwexasstitle{
5407 \protect\document@hwexamtype~\arabic{assignment}
5408 \assignment@title{}\;{}{}\; -- \given@due{}\}
5409 }

```

```

5410 \ifmultiple
5411 \newenvironment{@assignment}{
5412 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5413 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5414 }{
5415 \begin{omgroup}{\__hwexasstitle}
5416 }
5417 }{
5418 \end{omgroup}
5419 }

```

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

```

5420 \else
5421 \newenvironment{@assignment}{
5422 \begin{center}\bf
5423 \Large\@title\strut\
5424 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}\}
5425 \large\given@due{--\;}{\;}{--}
5426 \end{center}
5427 }{}
5428 \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.

```

5429 \keys_define:nn { hwexam / inclassignment } {
5430 %id .str_set_x:N = \l__hwexam_assign_id_str,
5431 number .int_set:N = \l__hwexam_inclassign_number_int,
5432 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5433 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5434 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5435 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5436 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5437 }
5438 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5439 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5440 \tl_clear:N \l__hwexam_inclassign_title_tl
5441 \tl_clear:N \l__hwexam_inclassign_type_tl
5442 \tl_clear:N \l__hwexam_inclassign_given_tl
5443 \tl_clear:N \l__hwexam_inclassign_due_tl
5444 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5445 \keys_set:nn { hwexam / inclassignment }{ #1 }
5446 }
5447 \__hwexam_inclassignment_args:n {}
5448
5449 \newcommand\inputassignment[2][ ]{
5450 \__hwexam_inclassignment_args:n { #1 }
5451 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5452 \input{#2}
5453 }{
5454 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5455 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5456 }
5457 }
5458 \__hwexam_inclasssign_assignment_args:n {}
5459 }
5460 \newcommand\includeassignment[2][ ]{
5461 \newpage
5462 \inputassignment[#1]{#2}
5463 }

```

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

35.4 Typesetting Exams

\quizheading

```

5464 \ExplSyntaxOff
5465 \newcommand\quizheading[1]{%
5466 \def\@tas{#1}%
5467 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5468 \ifx\@tas\empty\else%
5469 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5470 \fi%
5471 }
5472 \ExplSyntaxOn

```

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

\testheading

```

5473 \keys_define:nn { hwexam / testheading } {
5474 min .tl_set:N = \l__hwexam_testheading_min_tl,
5475 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5476 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5477 }
5478 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5479 \tl_clear:N \l__hwexam_testheading_min_tl
5480 \tl_clear:N \l__hwexam_testheading_duration_tl
5481 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5482 \keys_set:nn { hwexam / testheading }{ #1 }
5483 }
5484 \newenvironment{testheading}[1][ ]{
5485 \__hwexam_testheading_args:n{ #1 }
5486 \noindent\large{Name:~\hfill
5487 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5488 \begin{center}
5489 \Large\textbf{\@title}\[1ex]
5490 \large\@date\[3ex]
5491 \end{center}
5492 \textbf{You~have~
5493 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5494 \l__hwexam_testheading_min_tl~minutes
5495 }{
5496 \l__hwexam_testheading_duration_tl
5497 }~

```

```

5498 (sharp)~for~the~test
5499 };\
5500 Write~the~solutions~to~the~sheet.
5501 \par\noindent
5502 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5503 \advance\check@time by -\theassignment@totalmin
5504 The~estimated~time~for~solving~this~exam~is~
5505 {\theassignment@totalmin}~minutes,~
5506 leaving~you~{\the\check@time}~minutes~for~revising~
5507 your~exam.
5508
5509 \par\noindent
5510 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5511 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5512 You~can~reach~{\theassignment@totalpts}~points~if~you~
5513 solve~all~problems.~You~will~only~need~
5514 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5515 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
5516 \vfill
5517 \begin{center}
5518 {
5519 \Large\em You~have~ample~time,~so~take~it~slow~
5520 and~avoid~rushing~to~mistakes!\}[2ex]
5521 Different~problems~test~different~skills~and~
5522 knowledge,~so~do~not~get~stuck~on~one~problem.
5523 }
5524 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5525 \end{center}
5526 }{
5527 \newpage
5528 }

```

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

\testspace

```

5529 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5530 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5531 \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.

```

5532 <@=problems>
5533 \renewcommand\@problem[3]{
5534 \stepcounter{assignment@probs}
5535 \def\__problemspts{#2}

```

```

5536 \ifx\__problemspts\@empty\else
5537 \addtocounter{assignment@totalpts}{#2}
5538 \fi
5539 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5540 \xdef\correction@probs{\correction@probs & #1}%
5541 \xdef\correction@pts{\correction@pts & #2}
5542 \xdef\correction@reached{\correction@reached & }
5543 }
5544 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5545 \newcounter{assignment@probs}
5546 \newcounter{assignment@totalpts}
5547 \newcounter{assignment@totalmin}
5548 \def\correction@probs{\correction@probs@kw}%
5549 \def\correction@pts{\correction@pts@kw}%
5550 \def\correction@reached{\correction@reached@kw}%
5551 \def\after@correction@table{}%
5552 \stepcounter{assignment@probs}
5553 \newcommand\correction@table{
5554 \resizebox{\textwidth}{!}{%
5555 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5556 &\multicolumn{\theassignment@probs}{c|}||%|
5557 {\footnotesize\correction@forgrading@kw} &\\ \hline
5558 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5559 \correction@pts & \theassignment@totalpts & \\ \hline
5560 \correction@reached & & \[.7cm]\hline
5561 \end{tabular}}
5562 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5563 \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\wierfont=../assignments/wierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\wierglas{{\wierfont\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{\wierglas}

```