

The \TeX 3 Package *

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

2022-01-12

Abstract

TODO

*Version 3.0 (last revised 2022-01-12)

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	67
19.1	Generic Path Handling	67
19.2	PWD and kpsewhich	69
19.3	File Hooks and Tracking	70
19.4	MathHub Repositories	71
20	gTeX-References Implementation	77
20.1	Document URIs and URLs	77
20.2	Setting Reference Targets	79
20.3	Using References	80
21	gTeX-Modules Implementation	82
21.1	The module environment	85
21.2	Invoking modules	90
22	gTeX-Module Inheritance Implementation	92
22.1	SMS Mode	92
22.2	Inheritance	96

23	STEX-Symbols Implementation	101
23.1	Symbol Declarations	101
23.2	Notations	107
24	STEX-Terms Implementation	115
24.1	Symbol Invocations	115
24.2	Terms	118
24.3	Notation Components	124
25	STEX-Structural Features Implementation	127
25.1	The feature environment	127
25.2	Features	129
26	STEX-Statements Implementation	134
26.1	Definitions	135
26.2	Assertions	137
26.3	Examples	139
26.4	OMText	140
27	The Implementation	141
27.1	Package Options	141
27.2	Proofs	141
27.3	Justifications	147
28	STEX-Others Implementation	149
29	STEX-Metatheory Implementation	150
30	Tikzinput Implementation	153
31	document-structure.sty Implementation	155
31.1	The OMDoc Class	155
31.2	Class Options	155
31.3	Beefing up the document environment	156
31.4	Implementation: OMDoc Package	156
31.5	Package Options	156
31.6	Document Structure	158
31.7	Front and Backmatter	161
31.8	Global Variables	163
32	MiKoSlides – Implementation	164
32.1	Class and Package Options	164
32.2	Notes and Slides	166
32.3	Header and Footer Lines	170
32.4	Frame Images	171
32.5	Colors and Highlighting	172
32.6	Sectioning	173
32.7	Excursions	175

33 The Implementation	177
33.1 Package Options	177
33.2 Problems and Solutions	178
33.3 Multiple Choice Blocks	183
33.4 Including Problems	184
33.5 Reporting Metadata	185
34 Implementation: The hwexam Class	187
34.1 Class Options	187
35 Implementation: The hwexam Package	189
35.1 Package Options	189
35.2 Assignments	190
35.3 Including Assignments	193
35.4 Typesetting Exams	194
35.5 Leftovers	196

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

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

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

Example 7

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

$\textcolor{teal}{Multiplication}$ (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 (heuristically guessed) full path to the main file, based on the PWD and <code>\jobname</code> .
<code>\c_stex_pwd_str</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> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

.

5.1.1 The module-environment

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

`\stex_module_setup:nn` `\stex_module_setup:nn{\langle params \rangle}{\langle name \rangle}`
 Sets up a new module with name $\langle name \rangle$ and optional parameters $\langle params \rangle$. In particular, sets `\l_stex_current_module_prop` appropriately.

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

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

Test 4

```

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

```

```

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

```

.

Test 5

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

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

`\STEXModule` `\STEXModule {⟨fragment⟩}`

Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

Invoked by `\STEXModule`. Needs to be followed either by `!⟨macro⟩` or `?{⟨symbolname⟩}`. In the first case, it stores the full URI in `⟨macro⟩`; in the second case, it invokes the symbol `⟨symbolname⟩` in the selected module.

Test 6

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




`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

“SMS Mode” is used when loading modules from external tex files. It deactivates any output and ignores all T_EX commands not explicitly allowed via the following lists:

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

The names of environments that should be allowed in SMS mode. The corresponding `\begin`-statements are treated like the macros in `\g_stex_smsmode_allowedmacros_escape_tl`, so `\stex_smsmode_set_codes:` should be called at the end of the `\begin`-code. Since `\end`-statements take no arguments anyway, those are called with the SMS mode category code scheme active.

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

6.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

Module 6.1.1[Foo]

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

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

Module 6.1.2[Importtest]

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

Module 6.1.3[Importtest2]

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

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

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

Test 9

```

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

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

```

Module 6.1.4[UseTest1]

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

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

All modules: http://mathhub.info/sTeX?Metatheory, file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest3,
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2
All symbols: http://mathhub.info/sTeX?Metatheory?isa, http://mathhub.info/sTeX?Metatheory?bind, http://mathhub.info/sTeX?Metatheory?collect, http://mathhub.info/sTeX?Metatheory?fromto, http://mathhub.info/sTeX?Metatheory?apply, http://mathhub.info/sTeX?Metatheory?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}_{\com
```

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! [addition]</code> 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 \backslashwithbrackets .
<hr/> <hr/>	\backslashwithbrackets	$\backslashwithbrackets \langle left \rangle \langle right \rangle \{ \langle body \rangle \}$ 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 \backslashleft and \backslashright 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}
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\bar{plus}\{a,\mult\{b,c\}\}$  and  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
\displaystyle \plus{a,\mult{b,c}} and
\displaystyle \mult{a,\plus{\frac{ab}{c}}}
\withbrackets[]{$\displaystyle \plus{a,\mult{b,c}}$ and
\displaystyle \mult{a,\plus{\frac{ab}{c}}}$}
\end{module}

```

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

`\stex_term_custom:nn`

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

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

Test 16

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

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

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

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

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

\bar![bar]

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

\end{module}
```

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

`\stex_highlight_term:nn`

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

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

`\comp`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

Exports its argument as OMDOC (invisible), but does not produce PDF output. Useful e.g. for semantic macros that take arguments that are not part of the symbolic notation.

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

mathstructure TODO

Test 17

```

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 9.1.1[StructureTest1]

a**b**:*M*

file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma

feature?op

»macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}<

Test: *a*+*b*

Test2: *(U,+)*

30

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 spfcases environments that mark up the cases one by one.
spfcases	The content of a pfcases environment are a sequence of case proofs marked up in the pfcases environment, which takes an optional KeyVal argument for semantic annotations. The second argument is used to specify the the description of the case under consideration. The content of a pfcases environment is the same as that of a proof , i.e. steps , proofcomments , and pfcases environments. \spfcasesketch is a variant of the spfcases environment that takes the same arguments, but instead of the spfsteps in the body uses a third argument for a proof sketch.
\spfcasesketch	
sproofcomment	The proofcomment environment is much like a step , only that it does not have an object-level assertion of its own. Rather than asserting some fact that is relevant for the proof, it is used to explain where the proof is going, what we are attempting to to, or what we have achieved so far. As such, it cannot be the target of a \premise .

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

Chapter 12

sTeX-Metatheory

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

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

12.1 Symbols

Part III
Extensions

Chapter 13

Tikzinput

13.1 Macros and Environments

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

Chapter 14

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

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

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

14.1 Introduction

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

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

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

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 \LaTeX packages

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

14.2.2 Document Structure

document

\documentkeys

id

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

omgroup

id

creators

contributors

short

loadmodules

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

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

blindomgroup

\LaTeX 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`[$\langle URL \rangle$]{ $\langle label \rangle$ }, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL $\langle URL \rangle$ that lets \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`{ $\langle vname \rangle$ }{ $\langle text \rangle$ } to set the global variable $\langle vname \rangle$ to $\langle text \rangle$ and `\useSGvar`{ $\langle vname \rangle$ } to reference it.

With `\ifSGvar` we can test for the contents of a global variable: the macro call `\ifSGvar`{ $\langle vname \rangle$ }{ $\langle val \rangle$ }{ $\langle ctext \rangle$ } tests the content of the global variable $\langle vname \rangle$, only if (after expansion) it is equal to $\langle val \rangle$, the conditional text $\langle ctext \rangle$ 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.

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.

Name:
MatriculationNumber:

2022-01-12

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

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

[illegible]

Example 8: A generated test heading.

Part IV

Implementation

Chapter 18

STEX -Basics Implementation

18.1 The STEXDocument Class

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

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

18.2 Preliminaries

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

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

18.3 Messages and logging

```

40 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

18.4 Persistence

75 $\langle @@=stex_persist \rangle$

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

```

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

```

(End definition for $\backslash c_stex_persist_sms_iow$.)

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

```

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

```

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

18.5 HTML Annotations

```

94  $\langle @@=stex\_annotate \rangle$ 
95 \RequirePackage{rustex}

```

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

```

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

```

(End definition for __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool Whether to (locally) produce HTML output

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

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

```

(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

```

219 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

```

231
232 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
233   english   = en ,
234   ngerman   = de ,
235   arabic    = ar ,
236   bulgarian = bg ,
237   russian   = ru ,
238   finnish   = fi ,
239   romanian  = ro ,
240   turkish   = tr ,
241   french    = fr
242 }
243 % todo: chinese simplified (zhs)
244 %       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:

```

245 \clist_if_empty:NF \c_stex_languages_clist {
246   \clist_clear:N \l_tmpa_clist
247   \clist_map_inline:Nn \c_stex_languages_clist {
248     \prop_get:NnNTF \c_stex_languages_prop { #1 } \l_tmpa_str {
249       \clist_put_right:No \l_tmpa_clist \l_tmpa_str
250     } {
251       \msg_error:nxx{stex}{error/unknownlanguage}{\l_tmpa_str}
252     }
253   }
254   \stex_debug:nn{lang} {Languages:~\clist_use:Nn \l_tmpa_clist {,~} }
255   \RequirePackage[\clist_use:Nn \l_tmpa_clist,]{babel}
256 }

```

18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

257 \cs_new_protected:Nn \stex_deactivate_macro:Nn {
258   \exp_after:wN\let\csname \detokenize{#1} - orig\endcsname#1
259   \def#1{
260     \msg_error:nnnn{stex}{error/deactivated-macro}{#1}{#2}
261   }
262 }

```

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

`\stex_reactivate_macro:N`

```

263 \cs_new_protected:Nn \stex_reactivate_macro:N {
264   \exp_after:wN\let\exp_after:wN#1\csname \detokenize{#1} - orig\endcsname
265 }

```

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

```

266 </package>

```


Chapter 19

STEX -MathHub Implementation

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

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

```

293     }
294     \seq_set_eq:NN #1 \l_tmpa_tl
295   }
296   \stex_path_canonicalize:N #1
297 }
298 }
299 \cs_generate_variant:Nn \stex_path_from_string:Nn
300 { 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`

```

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

```

(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`
`\c__stex_path_up_str`

. and .., respectively.

```

308 \str_const:Nn \c__stex_path_dot_str {.}
309 \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.

```

310 \cs_new_protected:Nn \stex_path_canonicalize:N {
311   \seq_if_empty:NF #1 {
312     \seq_clear:N \l_tmpa_seq
313     \seq_get_left:NN #1 \l_tmpa_tl
314     \str_if_empty:NT \l_tmpa_tl {
315       \seq_put_right:Nn \l_tmpa_seq {}
316     }
317     \seq_map_inline:Nn #1 {
318       \str_set:Nn \l_tmpa_tl { ##1 }
319       \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_dot_str {} {
320         \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
321           \seq_if_empty:NTF \l_tmpa_seq {
322             \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
323               \c__stex_path_up_str
324             }
325           }{
326             \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
327             \str_if_eq:NNTF \l_tmpa_tl \c__stex_path_up_str {
328               \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
329                 \c__stex_path_up_str
330               }
331             }{
332               \seq_pop_right:NN \l_tmpa_seq \l_tmpb_tl
333             }

```

```

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

```

(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`

```

345 \prg_new_conditional:Nnn \stex_path_if_absolute:N {p, T, F, TF} {
346   \seq_if_empty:NTF #1 {
347     \prg_return_false:
348   }{
349     \seq_get_left:NN #1 \l_tmpa_tl
350     \str_if_empty:NTF \l_tmpa_tl {
351       \prg_return_true:
352     }{
353       \prg_return_false:
354     }
355   }
356 }

```

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

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

357 \str_new:N\l_stex_kpsewhich_return_str
358 \cs_new_protected:Nn \stex_kpsewhich:n {
359   \sys_get_shell:nnN { kpsewhich ~ #1 } { } \l_tmpa_tl
360   \exp_args:NNo\str_set:Nn\l_stex_kpsewhich_return_str{\l_tmpa_tl}
361   \tl_trim_spaces:N \l_stex_kpsewhich_return_str
362 }

```

(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`

```

363 \sys_if_platform_windows:TF{
364   \stex_kpsewhich:n{-expand-var~\c_percent_str CD\c_percent_str}
365 }{
366   \stex_kpsewhich:n{-var-value~PWD}
367 }
368
369 \stex_path_from_string:Nn\c_stex_pwd_seq\l_stex_kpsewhich_return_str
370 \stex_path_to_string:NN\c_stex_pwd_seq\c_stex_pwd_str
371 \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

372 `<@@=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

373 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

375 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

376 `\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`.

```

377 \seq_gclear_new:N\g_stex_currentfile_seq
378 \AddToHook{file/before}{
379   \stex_path_from_string:Nn\g_stex_currentfile_seq{\CurrentFilePath}
380   \stex_path_if_absolute:NTF\g_stex_currentfile_seq{
381     \exp_args:NNe\seq_put_right:Nn\g_stex_currentfile_seq{\CurrentFile}
382   }{
383     \stex_path_from_string:Nn\g_stex_currentfile_seq{
384       \c_stex_pwd_str/\CurrentFilePath/\CurrentFile
385     }
386   }
387   \seq_gset_eq:NN\g_stex_currentfile_seq\g_stex_currentfile_seq
388   \exp_args:NNo\seq_gpush:Nn\g__stex_files_stack\g_stex_currentfile_seq
389 }
390 \AddToHook{file/after}{
391   \seq_if_empty:NF\g__stex_files_stack{
392     \seq_gpop:NN\g__stex_files_stack\l_tmpa_seq
393   }
394   \seq_if_empty:NTF\g__stex_files_stack{
395     \seq_gset_eq:NN\g_stex_currentfile_seq\c_stex_mainfile_seq
396   }{
397     \seq_get:NN\g__stex_files_stack\l_tmpa_seq
398     \seq_gset_eq:NN\g_stex_currentfile_seq\l_tmpa_seq
399   }
400 }
```

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

19.4 MathHub Repositories

```

401 <@@=stex_mathhub>

\mathhub
\c_stex_mathhub_seq
\c_stex_mathhub_str
402 \str_if_empty:NTF\mathhub{
403   \stex_kpsewhich:n{-var-value~MATHHUB}
404   \str_set_eq:NN\c_stex_mathhub_str\l_stex_kpsewhich_return_str
405
406   \str_if_empty:NTF\c_stex_mathhub_str{
407     \msg_warning:nn{stex}{warning/nomathhub}
408   }{
409     \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
410     \exp_args:NNo \stex_path_from_string:Nn\c_stex_mathhub_seq\c_stex_mathhub_str
411   }
412 }{
413   \stex_path_from_string:Nn \c_stex_mathhub_seq \mathhub
414   \stex_path_if_absolute:NF \c_stex_mathhub_seq {
415     \exp_args:NNx \stex_path_from_string:Nn \c_stex_mathhub_seq {
416       \c_stex_pwd_str/\mathhub
417     }
418   }
419   \stex_path_to_string:NN\c_stex_mathhub_seq\c_stex_mathhub_str
420   \stex_debug:nn{mathhub} {MathHub:~\str_use:N\c_stex_mathhub_str}
421 }

```

(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
422 \cs_new_protected:Nn \__stex_mathhub_do_manifest:n {
423   \str_set:Nx \l_tmpa_str { #1 }
424   \prop_if_exist:cF {c_stex_mathhub_#1_manifest_prop} {
425     \prop_new:c { c_stex_mathhub_#1_manifest_prop }
426     \seq_set_split:NnV \l_tmpa_seq / \l_tmpa_str
427     \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpa_seq
428     \__stex_mathhub_find_manifest:N \l_tmpa_seq
429     \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
430       \msg_error:nnnn{stex}{error/norepository}{#1}{
431         \stex_path_to_string:N \c_stex_mathhub_str
432       }
433     } {
434       \exp_args:No \__stex_mathhub_parse_manifest:n { \l_tmpa_str }
435     }
436   }
437 }

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
438 \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`:

```

439 \cs_new_protected:Nn \_stex_mathhub_find_manifest:N {
440   \seq_set_eq:NN \l_tmpa_seq #1
441   \bool_set_true:N \l_tmpa_bool
442   \bool_while_do:Nn \l_tmpa_bool {
443     \seq_if_empty:NTF \l_tmpa_seq {
444       \bool_set_false:N \l_tmpa_bool
445     }{
446       \file_if_exist:nTF{
447         \stex_path_to_string:N \l_tmpa_seq/MANIFEST.MF
448       }{
449         \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
450         \bool_set_false:N \l_tmpa_bool
451       }{
452         \file_if_exist:nTF{
453           \stex_path_to_string:N \l_tmpa_seq/META-INF/MANIFEST.MF
454         }{
455           \seq_put_right:Nn \l_tmpa_seq{META-INF}
456           \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
457           \bool_set_false:N \l_tmpa_bool
458         }{
459           \file_if_exist:nTF{
460             \stex_path_to_string:N \l_tmpa_seq/meta-inf/MANIFEST.MF
461           }{
462             \seq_put_right:Nn \l_tmpa_seq{meta-inf}
463             \seq_put_right:Nn \l_tmpa_seq{MANIFEST.MF}
464             \bool_set_false:N \l_tmpa_bool
465           }{
466             \seq_pop_right:NN \l_tmpa_seq \l_tmpa_tl
467           }
468         }
469       }
470     }
471   }
472   \seq_set_eq:NN \l__stex_mathhub_manifest_file_seq \l_tmpa_seq
473 }

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

474 \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:

```

475 \cs_new_protected:Nn \_stex_mathhub_parse_manifest:n {
476   \seq_set_eq:NN \l_tmpa_seq \l__stex_mathhub_manifest_file_seq
477   \ior_open:Nn \c_stex_mathhub_manifest_ior {\stex_path_to_string:N \l_tmpa_seq}
478   \ior_map_inline:Nn \c_stex_mathhub_manifest_ior {
479     \str_set:Nn \l_tmpa_str {##1}
480     \exp_args:NNoo \seq_set_split:Nnn
481       \l_tmpb_seq \c_colon_str \l_tmpa_str
482     \seq_pop_left:NNTF \l_tmpb_seq \l_tmpa_tl {

```

```

483 \exp_args:NNe \str_set:Nn \l_tmpb_tl {
484 \exp_args:NNo \seq_use:Nn \l_tmpb_seq \c_colon_str
485 }
486 \exp_args:No \str_case:nnTF \l_tmpa_tl {
487 {id} {
488 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
489 { id } \l_tmpb_tl
490 }
491 {narration-base} {
492 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
493 { narr } \l_tmpb_tl
494 }
495 {url-base} {
496 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
497 { docurl } \l_tmpb_tl
498 }
499 {source-base} {
500 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
501 { ns } \l_tmpb_tl
502 }
503 {ns} {
504 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
505 { ns } \l_tmpb_tl
506 }
507 {dependencies} {
508 \prop_gput:cno { c_stex_mathhub_#1_manifest_prop }
509 { deps } \l_tmpb_tl
510 }
511 }{}{}
512 }{}
513 }
514 \ior_close:N \c__stex_mathhub_manifest_ior
515 }

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

516 \cs_new_protected:Nn \stex_set_current_repository:n {
517 \stex_require_repository:n { #1 }
518 \prop_set_eq:Nc \l_stex_current_repository_prop {
519 c_stex_mathhub_#1_manifest_prop
520 }
521 }

```

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

`\stex_require_repository:n`

```

522 \cs_new_protected:Nn \stex_require_repository:n {
523 \prop_if_exist:cF { c_stex_mathhub_#1_manifest_prop } {
524 \stex_debug:nn{mathhub}{Opening~archive:~#1}
525 \__stex_mathhub_do_manifest:n { #1 }
526 \exp_args:Nx \stex_add_to_sms:n {
527 \prop_const_from_keyval:cn { c_stex_mathhub_#1_manifest_prop } {
528 id = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { id } ,
529 ns = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { ns } ,

```

```

530     narr = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { narr } ,
531     deps = \prop_item:cn { c_stex_mathhub_#1_manifest_prop } { deps }
532   }
533 }
534 }
535 }

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

536 \prop_new:N \l_stex_current_repository_prop
537
538 \__stex_mathhub_find_manifest:N \c_stex_pwd_seq
539 \seq_if_empty:NTF \l__stex_mathhub_manifest_file_seq {
540   \stex_debug:nn{mathhub}{Not~currently~in~a~MathHub~repository}
541 } {
542   \__stex_mathhub_parse_manifest:n { main }
543   \prop_get:Nn \c_stex_mathhub_main_manifest_prop {id}
544   \l_tmpa_str
545   \prop_set_eq:cN { c_stex_mathhub_ \l_tmpa_str _manifest_prop }
546   \c_stex_mathhub_main_manifest_prop
547   \exp_args:Nx \stex_set_current_repository:n { \l_tmpa_str }
548   \stex_debug:nn{mathhub}{Current~repository:~
549   \prop_item:Nn \l_stex_current_repository_prop {id}
550 }
551 }

```

(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.

```

552 \cs_new_protected:Nn \stex_in_repository:nn {
553   \str_set:Nx \l_tmpa_str { #1 }
554   \cs_set:Npn \l_tmpa_cs ##1 { #2 }
555   \str_if_empty:NTF \l_tmpa_str {
556     \exp_args:Ne \l_tmpa_cs{
557       \prop_item:Nn \l_stex_current_repository_prop { id }
558     }
559   }{
560     \stex_require_repository:n \l_tmpa_str
561     \str_set:Nx \l_tmpa_str { #1 }
562     \exp_args:Nne \use:nn {
563       \stex_set_current_repository:n \l_tmpa_str
564       \exp_args:Nx \l_tmpa_cs{\l_tmpa_str}
565     }{
566       \stex_set_current_repository:n {
567         \prop_item:Nn \l_stex_current_repository_prop { id }
568       }
569     }
570   }
571 }

```

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


```

\inputref
\stex_inputref:nn
572 \newif \ifinputref \inputreffalse
573
574 \cs_new_protected:Nn \stex_inputref:nn {
575   \stex_in_repository:nn {#1} {
576     \ifinputref
577       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
578     \else
579       \inputreftrue
580       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
581     \inputreffalse
582   \fi
583 }
584 }
585 \NewDocumentCommand \inputref { 0{} m}{
586   \stex_inputref:nn{ #1 }{ #2 }
587 }
588
589 \cs_new_protected:Nn \stex_mhbibresource:nn {
590   \stex_in_repository:nn {#1} {
591     \addbibresource{ \c_stex_mathhub_str / ##1 / #2 }
592   }
593 }
594 \newcommand\addmhbibresource[2][]{
595   \stex_mhbibresource:nn{ #1 }{ #2 }
596 }

```

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

`\mhpath`

```

597 \def \mhpath #1 #2 {
598   \exp_args:Ne \str_if_eq:nnTF{#1}{#{
599     \c_stex_mathhub_str /
600     \prop_item:Nn \l_stex_current_repository_prop { id }
601     / source / #2
602   }{
603     \c_stex_mathhub_str / #1 / source / #2
604   }
605 }

```

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

`\libinput`

```

606 \cs_new_protected:Npn \libinput #1 {
607   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
608     \msg_error:nnn{stex}{error/notinarchive}\libinput
609   }
610   \bool_set_false:N \l_tmpa_bool
611   \tl_clear:N \l_tmpa_tl
612   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
613   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
614   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
615   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
616     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

617 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
618 / meta-inf / lib / #1.tex}{
619 \bool_set_true:N \l_tmpa_bool
620 \tl_put_right:Nx \l_tmpa_tl {
621 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
622 / meta-inf / lib / #1.tex}
623 }
624 }{}
625 }
626 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
627 / \l_tmpa_str / lib / #1.tex
628 }{
629 \bool_set_true:N \l_tmpa_bool
630 \tl_put_right:Nx \l_tmpa_tl {
631 \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
632 / \l_tmpa_str / lib / #1.tex}
633 }
634 }{}
635 \bool_if:NF \l_tmpa_bool {
636 \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
637 }
638 \l_tmpa_tl
639 }

```

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

```

640 </package>

```

Chapter 20

STEX -References Implementation

```
641 <*package>
642
643 %%%%%%%%%% references.dtx %%%%%%%%%%
644
645 %\RequirePackage{hyperref}
646 %\RequirePackage{cleveref}
647 <@@=stex_refs>
648
649 Warnings and error messages
650
651 \iow_new:N \c__stex_refs_refs_iow
652 \AddToHook{begindocument}{
653   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
654 }
655 \AddToHook{enddocument}{
656   \iow_close:N \c__stex_refs_refs_iow
657 }
658
659 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
660
661 \NewDocumentCommand \STEXreftitle { m } {
662   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
663 }
664
```

20.1 Document URIs and URLs

```
662 \seq_new:N \g__stex_refs_all_refs_seq
663
664 \str_new:N \l_stex_current_docns_str
665
666 \cs_new_protected:Nn \stex_get_document_uri: {
667   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
668   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
669   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
670   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
671 }
672
```

```

671 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
672
673 \str_clear:N \l_tmpa_str
674 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
675   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
676 }
677
678 \str_if_empty:NTF \l_tmpa_str {
679   \str_set:Nx \l_stex_current_docns_str {
680     file:/\stex_path_to_string:N \l_tmpa_seq
681   }
682 }{
683   \bool_set_true:N \l_tmpa_bool
684   \bool_while_do:Nn \l_tmpa_bool {
685     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
686     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
687       {source} { \bool_set_false:N \l_tmpa_bool }
688     }{}{
689       \seq_if_empty:NT \l_tmpa_seq {
690         \bool_set_false:N \l_tmpa_bool
691       }
692     }
693   }
694
695   \seq_if_empty:NTF \l_tmpa_seq {
696     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
697   }{
698     \str_set:Nx \l_stex_current_docns_str {
699       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
700     }
701   }
702 }
703 }
704
705 \str_new:N \l_stex_current_docurl_str
706 \cs_new_protected:Nn \stex_get_document_url: {
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   \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 { docurl } \l_tmpa_str {
715     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
716       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
717     }
718   }
719
720   \str_if_empty:NTF \l_tmpa_str {
721     \str_set:Nx \l_stex_current_docurl_str {
722       file:/\stex_path_to_string:N \l_tmpa_seq
723     }
724   }{
725     \bool_set_true:N \l_tmpa_bool

```

```

725 \bool_while_do:Nn \l_tmpa_bool {
726   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
727   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
728     {source} { \bool_set_false:N \l_tmpa_bool }
729   }{}{
730     \seq_if_empty:NT \l_tmpa_seq {
731       \bool_set_false:N \l_tmpa_bool
732     }
733   }
734 }
735
736 \seq_if_empty:NTF \l_tmpa_seq {
737   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
738 }{
739   \str_set:Nx \l_stex_current_docurl_str {
740     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
741   }
742 }
743 }
744 }

```

20.2 Setting Reference Targets

```

745 \str_const:Nn \c__stex_refs_url_str{URL}
746 \str_const:Nn \c__stex_refs_ref_str{REF}
747 % @currentlabel -> number
748 % @currentlabelname -> title
749 % @currentHref -> name.number <- id of some kind
750 % \theH# -> \arabic{section}
751 % \the# -> number
752 % \hyper@makecurrent{#}
753 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
754   \stex_get_document_uri:
755   \str_set:Nx \l_tmpa_str { #1 }
756   \str_if_empty:NT \l_tmpa_str {
757     \int_zero:N \l_tmpa_int
758     \bool_set_true:N \l_tmpa_bool
759     \bool_while_do:Nn \l_tmpa_bool {
760       \cs_if_exist:cTF {
761         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
762       }{
763         \int_incr:N \l_tmpa_int
764       }{
765         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
766         \bool_set_false:N \l_tmpa_bool
767       }
768     }
769   }
770   \str_set:Nx \l_tmpa_str {
771     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
772   }
773   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
774   \stex_if_smsmode:TF {
775     \stex_get_document_url:

```

```

776 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
777 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
778 }{
779 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~=\expandafter{\@currentlabel\iffalse}}
780 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
781 \str_gset:cn {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
782 }
783 }

784 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
785 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
786 }

```

20.3 Using References

```

787 \str_new:N \l__stex_refs_indocument_str
788 \keys_define:nn { stex / sref } {
789   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
790   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
791   pre           .tl_set:N = \l__stex_refs_pre_tl ,
792   post          .tl_set:N = \l__stex_refs_post_tl ,
793   %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
794 }
795
796 \bool_new:N \c__stex_refs_hyperref_bool
797 \bool_set_false:N \c__stex_refs_hyperref_bool
798 \AddToHook{begindocument}{
799   \@ifpackageloaded{hyperref}{
800     \bool_set_true:N \c__stex_refs_hyperref_bool
801   }{}
802 }
803
804
805 \cs_new_protected:Nn \__stex_refs_args:n {
806   \tl_clear:N \l__stex_refs_linktext_tl
807   \tl_clear:N \l__stex_refs_fallback_tl
808   \tl_clear:N \l__stex_refs_pre_tl
809   \tl_clear:N \l__stex_refs_post_tl
810   \str_clear:N \l__stex_refs_repo_str
811   \keys_set:nn { stex / sref } { #1 }
812 }
813
814 \NewDocumentCommand \sref { 0{} m }{
815   \__stex_refs_args:n { #1 }
816   \str_if_empty:NTF \l__stex_refs_indocument_str {
817     \str_set:Nn \l_tmpa_str { #2 }
818     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
819     \tl_set:Nn \l_tmpa_tl {
820       \l__stex_refs_fallback_tl
821     }
822     \seq_map_inline:Nn \g__stex_refs_all_refs_seq {
823       \str_set:Nn \l_tmpb_str { ##1 }
824       \str_if_eq:eeT { \l_tmpa_str } {
825         \str_range:Nnn \l_tmpb_str { -\l_tmpa_int }{-1 }
826       } {

```

```

827     \seq_map_break:n {
828         \tl_set:Nn \l_tmpa_tl {
829             % doc uri in \l_tmpb_str
830             \str_set:Nx \l_tmpa_str {sref_url_\l_tmpb_str_type}
831             \str_if_eq:NNTF \l_tmpa_str \c__stex_refs_ref_str {
832                 % reference
833                 \l__stex_refs_pre_tl\ref{sref_\l_tmpb_str}\l__stex_refs_post_tl
834             }{
835                 % URL
836                 \if_bool:N \c__stex_refs_hyperref_bool {
837                     \exp_args:Nx \href{\use:c{sref_url_\l_tmpb_str_str}}{\l__stex_refs_fallback
838                 }{
839                     \l__stex_refs_fallback_tl
840                 }
841             }
842         }
843     }
844 }
845 }
846 \l_tmpa_tl
847 }{
848     % TODO
849 }
850 }
851
852 </package>

```

Chapter 21

STEX -Modules Implementation

```
853 <*package>
854
855 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
856
857 <@@=stex_modules>
858
859 Warnings and error messages
858 \msg_new:nnn{stex}{error/unknownmodule}{
859   No~module~#1~found
860 }
861 \msg_new:nnn{stex}{error/syntax}{
862   Syntax~error:~#1
863 }
864 \msg_new:nnn{stex}{error/siglanguage}{
865   Module~#1~declares~signature~#2,~but~does~not~
866   declare~its~language
867 }
```

\l_stex_current_module_prop The current module:

```
868 \prop_new:N \l_stex_current_module_prop
```

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

\l_stex_all_modules_seq Stores all available modules

```
869 \seq_new:N \l_stex_all_modules_seq
```

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

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

\g_stex_module_files_prop

```
870 \seq_new:N \g_stex_modules_in_file_seq
871 \prop_new:N \g_stex_module_files_prop
```

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


```

\stex_if_in_module_p:
\stex_if_in_module:TF
872 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
873   \prop_if_empty:NTF \l_stex_current_module_prop
874   \prg_return_false: \prg_return_true:
875 }

```

(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
876 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
877   \prop_if_exist:cTF { c_stex_module_#1_prop }
878   \prg_return_true: \prg_return_false:
879 }

```

(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:
880 \cs_new_protected:Nn \stex_add_to_current_module:n {
881   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
882   \tl_put_right:Nn \l_tmpa_tl { #1 }
883   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
884 }
885 \cs_new_protected:Npn \STEXexport {
886   \begingroup
887   \newlinechar=-1\relax
888   \endlinechar=-1\relax
889   %\catcode'\ = 9\relax
890   \expandafter\endgroup\STEXexport:n
891 }
892 \cs_new_protected:Nn \STEXexport:n {
893   \ignorespaces #1
894   \stex_add_to_current_module:n { \ignorespaces #1 }
895   \stex_smsmode_set_codes:
896 }
897 \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
898 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
899   \str_set:Nx \l_tmpa_str { #1 }
900   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
901   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
902   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
903 }

```

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

```

\stex_add_import_to_current_module:n
904 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
905   \str_set:Nx \l_tmpa_str { #1 }
906   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
907   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
908   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
909 }

```

(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).

```

910 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
911   \str_set:Nx \l_tmpa_str { #1 }
912   \seq_set_eq:NN \l_tmpa_seq #2
913   % split off file extension
914   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
915   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
916   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
917   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
918
919   \bool_set_true:N \l_tmpa_bool
920   \bool_while_do:Nn \l_tmpa_bool {
921     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
922     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
923       {source} { \bool_set_false:N \l_tmpa_bool }
924     }{}{
925       \seq_if_empty:NT \l_tmpa_seq {
926         \bool_set_false:N \l_tmpa_bool
927       }
928     }
929   }
930
931   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
932   \str_if_empty:NTF \l_stex_modules_subpath_str {
933     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
934   }{
935     \str_set:Nx \l_stex_modules_ns_str {
936       \l_tmpa_str/\l_stex_modules_subpath_str
937     }
938   }
939 }
```

(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`

```

940 \str_new:N \l_stex_modules_ns_str
941 \str_new:N \l_stex_modules_subpath_str
```

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

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

```

942 \cs_new_protected:Nn \stex_modules_current_namespace: {
943   \str_clear:N \l_stex_modules_subpath_str
944   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
945     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
946   }{
947     % split off file extension
948     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
949     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
```

```

950     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
951     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
952     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
953     \str_set:Nx \l_stex_modules_ns_str {
954         file:/\stex_path_to_string:N \l_tmpa_seq
955     }
956 }
957 }

```

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

21.1 The module environment

module arguments:

```

958 \keys_define:nn { stex / module } {
959     title      .str_set_x:N = \l_stex_module_title_str ,
960     ns         .str_set_x:N = \l_stex_module_ns_str ,
961     lang       .str_set_x:N = \l_stex_module_lang_str ,
962     sig        .str_set_x:N = \l_stex_module_sig_str ,
963     creators   .str_set_x:N = \l_stex_module_creators_str ,
964     contributors .str_set_x:N = \l_stex_module_contributors_str ,
965     meta       .str_set_x:N = \l_stex_module_meta_str
966 }
967
968 \cs_new_protected:Nn \__stex_modules_args:n {
969     \str_clear:N \l_stex_module_title_str
970     \str_clear:N \l_stex_module_ns_str
971     \str_clear:N \l_stex_module_lang_str
972     \str_clear:N \l_stex_module_sig_str
973     \str_clear:N \l_stex_module_creators_str
974     \str_clear:N \l_stex_module_contributors_str
975     \str_clear:N \l_stex_module_meta_str
976     \keys_set:nn { stex / module } { #1 }
977 }
978
979 % module parameters here? In the body?
980

```

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

```

981 \cs_new_protected:Nn \stex_module_setup:nn {
982     \str_set:Nx \l_stex_module_name_str { #2 }
983     \__stex_modules_args:n { #1 }

```

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

```

984     \stex_if_in_module:TF {
985         % Nested module
986         \prop_get:NnN \l_stex_current_module_prop
987             { ns } \l_stex_module_ns_str
988         \str_set:Nx \l_stex_module_name_str {
989             \prop_item:Nn \l_stex_current_module_prop
990                 { name } / \l_stex_module_name_str
991         }

```

```

992 }{
993   % not nested:
994   \str_if_empty:NT \l_stex_module_ns_str {
995     \stex_modules_current_namespace:
996     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
997     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
998       / {\l_stex_module_ns_str}
999     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1000     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1001       \str_set:Nx \l_stex_module_ns_str {
1002         \stex_path_to_string:N \l_tmpa_seq
1003       }
1004     }
1005   }
1006 }

```

Next, we determine the language of the module:

```

1007 \str_if_empty:NT \l_stex_module_lang_str {
1008   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1009   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1010   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1011   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1012   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1013     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1014       inferred~from~file~name}
1015     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1016   }
1017 }
1018
1019 \str_if_empty:NF \l_stex_module_lang_str {
1020   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
1021     \l_tmpa_str {
1022     \ltx@ifpackageloaded{babel}{
1023       \exp_args:Nx \selectlanguage { \l_tmpa_str }
1024     }{}
1025   } {
1026     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
1027   }
1028 }

```

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

```

1029 \str_if_empty:NTF \l_stex_module_sig_str {
1030   \str_clear:N \l_tmpa_str
1031   \seq_clear:N \l_tmpa_seq
1032   \tl_clear:N \l_tmpa_tl
1033   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
1034     name      = \l_stex_module_name_str ,
1035     ns        = \l_stex_module_ns_str ,
1036     imports   = \exp_not:o { \l_tmpa_seq } ,
1037     constants = \exp_not:o { \l_tmpa_seq } ,
1038     content   = \exp_not:o { \l_tmpa_tl } ,
1039     file      = \exp_not:o { \g_stex_currentfile_seq } ,
1040     lang      = \l_stex_module_lang_str ,

```

```

1041     sig      = \l_stex_module_sig_str ,
1042     meta     = \l_stex_module_meta_str
1043   }
1044 }{
1045   \str_if_empty:NT \l_stex_module_lang_str {
1046     \msg_error:nnnn{stex}{error/siglanguage}{
1047       \l_stex_module_ns_str?\l_stex_module_name_str
1048     }{\l_stex_module_sig_str}
1049   }
1050
1051   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1052   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1053   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1054   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1055   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1056   \str_set:Nx \l_tmpa_str {
1057     \stex_path_to_string:N \l_tmpa_seq /
1058     \l_tmpa_str . \l_stex_module_sig_str .tex
1059   }
1060   \IfFileExists \l_tmpa_str {
1061     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1062       \seq_clear:N \l_stex_all_modules_seq
1063       \prop_clear:N \l_stex_current_module_prop
1064       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1065       \input { \l_tmpa_str }
1066     }
1067   }{
1068     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1069   }
1070   \stex_activate_module:n {
1071     \l_stex_module_ns_str ? \l_stex_module_name_str
1072   }
1073   \prop_set_eq:Nc \l_stex_current_module_prop {
1074     c_stex_module_
1075     \l_stex_module_ns_str ?
1076     \l_stex_module_name_str
1077     _prop
1078   }
1079 }

```

We load the metatheory:

```

1080   \str_if_empty:NT \l_stex_module_meta_str {
1081     \str_set:Nx \l_stex_module_meta_str {
1082       \c_stex_metatheory_ns_str ? Metatheory
1083     }
1084   }
1085   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1086     \exp_args:Nx \stex_add_to_current_module:n {
1087       \stex_activate_module:n {\l_stex_module_meta_str}
1088     }
1089     \stex_activate_module:n {\l_stex_module_meta_str}
1090   }
1091 }

```

(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}

```

1092 \cs_new_protected:Nn \_stex_modules_begin_module:nn {
1093   \stex_reactivate_macro:N \STEXexport
1094   \stex_reactivate_macro:N \importmodule
1095   \stex_reactivate_macro:N \symdecl
1096   \stex_reactivate_macro:N \notation
1097   \stex_reactivate_macro:N \symdef
1098   \stex_module_setup:nn{#1}{#2}
1099
1100   \stex_debug:nn{modules}{
1101     New~module:\\
1102     Namespace:~\l_stex_module_ns_str\\
1103     Name:~\l_stex_module_name_str\\
1104     Language:~\l_stex_module_lang_str\\
1105     Signature:~\l_stex_module_sig_str\\
1106     Metatheory:~\l_stex_module_meta_str\\
1107     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1108   }
1109
1110   \seq_put_right:Nx \l_stex_all_modules_seq {
1111     \l_stex_module_ns_str ? \l_stex_module_name_str
1112   }
1113
1114   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1115     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1116
1117   \stex_if_smsmode:TF {
1118     \stex_smsmode_set_codes:
1119   } {
1120     \begin{stex_annotate_env} {theory} {
1121       \l_stex_module_ns_str ? \l_stex_module_name_str
1122     }
1123
1124     \stex_annotate_invisible:nnn{header}{} {
1125       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1126       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1127       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1128         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1129       }
1130     }
1131   }
1132   % TODO: Inherit metatheory for nested modules?
1133 }
1134 \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}

```

1135 \cs_new_protected:Nn \_stex_modules_end_module: {
1136   \str_set:Nx \l_tmpa_str {
1137     c_stex_module_
1138     \prop_item:Nn \l_stex_current_module_prop { ns } ?

```

```

1139     \prop_item:Nn \l_stex_current_module_prop { name }
1140     _prop
1141   }
1142   %^^A \prop_new:c { \l_tmpa_str }
1143   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1144   \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1145 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1146 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1147 \NewDocumentEnvironment { @module } { 0 } { m } {
1148   \par
1149   \_stex_modules_begin_module:nn{#1}{#2}
1150 } {
1151   \_stex_modules_end_module:
1152   \stex_if_smsmode:TF {
1153     \exp_args:Nx \stex_add_to_sms:n {
1154       \prop_gset_from_keyval:cn {
1155         c_stex_module_
1156         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1157         \prop_item:Nn \l_stex_current_module_prop { name }
1158         _prop
1159       } {
1160         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1161         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1162         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1163         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1164         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1165         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1166         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1167         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1168         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1169       }
1170     }
1171   }{
1172     \end{stex_annotate_env}
1173   }
1174 }

```

\stex_modules_heading: Code for document headers

```

1175 \cs_if_exist:NTF \thesection {
1176   \newcounter{module}[section]
1177 }{
1178   \newcounter{module}
1179 }
1180
1181 \bool_if:NT \c_stex_showmods_bool {
1182   \latexml_if:F { \RequirePackage{mdframed} }
1183 }
1184
1185 \cs_new_protected:Nn \stex_modules_heading: {
1186   \stepcounter{module}

```

```

1187 \par
1188 \bool_if:NT \c_stex_showmods_bool {
1189   \noindent{\textbf{Module} ~
1190     \cs_if_exist:NT \thesection {\thesection.}
1191     \themodule ~ [\l_stex_module_name_str]
1192   }
1193   \str_if_empty:NTF \l_stex_module_title_str {
1194   }{
1195     \quad(\l_stex_module_title_str)\hfill
1196   }\par
1197 }
1198 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1199 % TODO
1200 \stex_ref_new_doc_target:n \l_stex_module_name_str
1201 }

```

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

Finally:

```

1202 \NewDocumentEnvironment { module } { 0 } { m } {
1203   \bool_if:NT \c_stex_showmods_bool {
1204     \begin{mdframed}
1205   }
1206   \begin{@module}[#1]{#2}
1207   \stex_modules_heading:
1208 }{
1209   \end{@module}
1210   \bool_if:NT \c_stex_showmods_bool {
1211     \end{mdframed}
1212   }
1213 }

```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1214 \NewDocumentCommand \STEXModule { m } {
1215   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1216   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1217   \tl_set:Nn \l_tmpa_tl {
1218     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1219   }
1220   \seq_map_inline:Nn \l_stex_all_modules_seq {
1221     \str_set:Nn \l_tmpb_str { ##1 }
1222     \str_if_eq:eeT { \l_tmpa_str } {
1223       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1224     } {
1225       \seq_map_break:n {
1226         \tl_set:Nn \l_tmpa_tl {
1227           \stex_invoke_module:n { ##1 }
1228         }
1229       }
1230     }
1231   }
1232   \l_tmpa_tl

```



```

1233 }
1234
1235 \cs_new_protected:Nn \stex_invoke_module:n {
1236   \stex_debug:nn{modules}{Invoking~module~#1}
1237   \peek_charcode_remove:NTF ! {
1238     \__stex_modules_invoke_uri:nN { #1 }
1239   } {
1240     \peek_charcode_remove:NTF ? {
1241       \__stex_modules_invoke_symbol:nn { #1 }
1242     } {
1243       \msg_error:nnn{stex}{error/syntax}{
1244         ?~or~!~expected~after~
1245         \c_backslash_str STEXModule{#1}
1246       }
1247     }
1248   }
1249 }
1250
1251 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1252   \str_set:Nn #2 { #1 }
1253 }
1254
1255 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1256   \stex_invoke_symbol:n{#1?#2}
1257 }

```

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

`\stex_activate_module:n`

```

1258 \cs_new_protected:Nn \stex_activate_module:n {
1259   \stex_debug:nn{modules}{Activating~module~#1}
1260   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1261     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1262     \prop_item:cn { c_stex_module_#1_prop } { content }
1263   }
1264 }

```

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

```

1265 </package>

```

Chapter 22

sTeX -Module Inheritance Implementation

```
1266 <*package>
1267
1268 %%%%%%%%% inheritance.dtx %%%%%%%%%
1269
```

22.1 SMS Mode

```
1270 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1271 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1272 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1273 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1274
1275 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1276   \makeatletter
1277   \makeatother
1278   \ExplSyntaxOn
1279   \ExplSyntaxOff
1280 }
1281
1282 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1283   \symdef
1284   \importmodule
1285   \notation
1286   \symdecl
1287   \STEXexport
1288 }
1289
1290 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1291   \tl_to_str:n {
1292     module,
1293     @module
```

```

1294 }
1295 }

```

(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

```

```

1296 \bool_new:N \g__stex_smsmode_bool
1297 \bool_set_false:N \g__stex_smsmode_bool
1298 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1299   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1300 }

```

(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

```

```

1301 \bool_new:N \g__stex_smsmode_catcode_bool
1302 \bool_set_false:N \g__stex_smsmode_catcode_bool
1303 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1304   \bool_if:NTF \g__stex_smsmode_catcode_bool
1305   \prg_return_true: \prg_return_false:
1306 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1307 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1308   \stex_if_smsmode:T {
1309     \__stex_smsmode_if_catcodes:F {
1310       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1311       \exp_after:wN \char_gset_active_eq:NN
1312       \c_backslash_str \__stex_smsmode_cs:
1313       \tex_global:D \char_set_catcode_active:N \
1314       \tex_global:D \char_set_catcode_other:N $
1315       \tex_global:D \char_set_catcode_other:N ^
1316       \tex_global:D \char_set_catcode_other:N _
1317       \tex_global:D \char_set_catcode_other:N &
1318       \tex_global:D \char_set_catcode_other:N ##
1319     }
1320   }
1321 } \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.

```

1322 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1323   \__stex_smsmode_if_catcodes:T {
1324     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1325     \exp_after:wN \tex_global:D \exp_after:wN
1326     \char_set_catcode_escape:N \c_backslash_str
1327     \tex_global:D \char_set_catcode_math_toggle:N $
1328     \tex_global:D \char_set_catcode_math_superscript:N ^
1329     \tex_global:D \char_set_catcode_math_subscript:N _
1330     \tex_global:D \char_set_catcode_alignment:N &
1331     \tex_global:D \char_set_catcode_parameter:N ##
1332   }
1333 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1334 \cs_new_protected:Nn \stex_in_smsmode:nn {
1335   \vbox_set:Nn \l_tmpa_box {
1336     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1337     \bool_gset_true:N \g__stex_smsmode_bool
1338     \stex_smsmode_set_codes:
1339     #2
1340     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1341     \stex_if_smsmode:F {
1342       \__stex_smsmode_unset_codes:
1343     }
1344   }
1345   \box_clear:N \l_tmpa_box
1346 }

```

(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:

```

1347 \cs_new_protected:Nn \_stex_smsmode_cs: {
1348   \str_clear:N \l_tmpa_str
1349   \peek_analysis_map_inline:n {
1350     % #1: token (one expansion)
1351     % #2: charcode
1352     % #3 catcode
1353     \token_if_eq_charcode:NNTF ##3 B {
1354       % token is a letter
1355       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1356     } {
1357       \str_if_empty:NTF \l_tmpa_str {
1358         % we don't allow (or need) single non-letter CSs
1359         % for now
1360         \peek_analysis_map_break:
1361       }{
1362         \str_if_eq:onTF \l_tmpa_str { begin } {
1363           \peek_analysis_map_break:n {
1364             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1365           }
1366         } {
1367           \str_if_eq:onTF \l_tmpa_str { end } {
1368             \peek_analysis_map_break:n {
1369               \exp_after:wN \_stex_smsmode_checkend:n ##1
1370             }
1371           } {
1372             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1373             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1374               \g_stex_smsmode_allowedmacros_tl
1375               { \use:c{\l_tmpa_str} } {
1376               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1377               \peek_analysis_map_break:n {
1378                 \exp_after:wN \l_tmpa_tl ##1
1379               }
1380             }

```

```

1380     } {
1381         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1382         \g_stex_smsmode_allowedmacros_escape_tl
1383         { \use:c{\l_tmpa_str} } {
1384             \__stex_smsmode_unset_codes:
1385             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1386             % TODO \__stex_smsmode_rescan_cs:
1387             \int_compare:nNnTF {##2} = {92} {
1388                 \peek_analysis_map_break:n {
1389                     \__stex_smsmode_unset_codes:
1390                     \__stex_smsmode_rescan_cs:
1391                 }
1392             } {
1393                 \peek_analysis_map_break:n {
1394                     \exp_after:wN \l_tmpa_tl ##1
1395                 }
1396             }
1397         } {
1398             \int_compare:nNnTF {##2} = {92} {
1399                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1400             } {
1401                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1402             }
1403         }
1404     }
1405 }
1406 }
1407 }
1408 }
1409 }
1410 }

```

(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:

```

1411 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1412     \str_clear:N \l_tmpb_str
1413     \peek_analysis_map_inline:n {
1414         \token_if_eq_charcode:NNTF ##3 B {
1415             % token is a letter
1416             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1417         } {
1418             \peek_analysis_map_break:n {
1419                 \exp_after:wN \use:c \exp_after:wN {
1420                     \exp_after:wN \l_tmpa_str\exp_after:wN
1421                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1422             }
1423         }
1424     }
1425 }

```

(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.

```

1426 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1427   \str_set:Nn \l_tmpa_str { #1 }
1428   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1429     \__stex_smsmode_unset_codes:
1430     \begin{#1}
1431   }
1432 }
```

(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.

```

1433 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1434   \str_set:Nn \l_tmpa_str { #1 }
1435   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1436     \end{#1}
1437   }
1438 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

1439 `<@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1440 \cs_new_protected:Nn \stex_import_module_uri:nn {
1441   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1442   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1443
1444   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1445   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1446   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1447
1448   \stex_modules_current_namespace:
1449   \bool_lazy_all:nTF {
1450     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1451     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1452     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1453   }{
1454     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1455     \str_set_eq:NN \l_stex_module_ns
1456   }{
1457     \str_if_empty:NT \l__stex_importmodule_archive_str {
1458       \prop_if_empty:NF \l_stex_current_repository_prop {
1459         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1460       }
1461     }
1462     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1463       \str_if_empty:NF \l__stex_importmodule_path_str {
1464         \str_set:Nx \l_stex_module_ns_str {
1465           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1466         }
1467       }
1468     }
```

```

1468   }{
1469     \stex_require_repository:n \l__stex_importmodule_archive_str
1470     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1471       \l_stex_module_ns_str
1472     \str_if_empty:NF \l__stex_importmodule_path_str {
1473       \str_set:Nx \l_stex_module_ns_str {
1474         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1475       }
1476     }
1477   }
1478 }
1479 }

```

(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>	1480 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1481 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1482 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1483 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1484 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1485   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1486
1487     % archive
1488     \str_set:Nx \l_tmpa_str { #2 }
1489     \str_if_empty:NTF \l_tmpa_str {
1490       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1491     } {
1492       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1493       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1494       \seq_put_right:Nn \l_tmpa_seq { source }
1495     }
1496
1497     % path
1498     \str_set:Nx \l_tmpb_str { #3 }
1499     \str_if_empty:NTF \l_tmpb_str {
1500       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1501
1502       \ltx@ifpackageloaded{babel} {
1503         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1504           { \language } \l_tmpb_str {
1505           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1506         }
1507       } {
1508         \str_clear:N \l_tmpb_str
1509       }
1510
1511       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1512       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1513         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1514 }{
1515   \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1516   \IfFileExists{ \l_tmpa_str.tex }{
1517     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1518   }{
1519     % try english as default
1520     \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1521     \IfFileExists{ \l_tmpa_str.en.tex }{
1522       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1523     }{
1524       \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1525     }
1526   }
1527 }
1528
1529 } {
1530   \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1531   \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1532
1533   \ltx@ifpackageloaded{babel} {
1534     \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1535       { \language } \l_tmpb_str {
1536         \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1537       }
1538   } {
1539     \str_clear:N \l_tmpb_str
1540   }
1541
1542   \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1543
1544   \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1545   \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1546     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1547   }{
1548     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1549     \IfFileExists{ \l_tmpa_str/#4.tex }{
1550       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1551     }{
1552       % try english as default
1553       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1554       \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1555         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1556       }{
1557         \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1558         \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1559           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1560         }{
1561           \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1562           \IfFileExists{ \l_tmpa_str.tex }{
1563             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1564           }{
1565             % try english as default
1566             \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1567             \IfFileExists{ \l_tmpa_str.en.tex }{

```



```

1568         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1569     }{
1570         \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1571     }
1572 }
1573 }
1574 }
1575 }
1576 }
1577 }
1578
1579 \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1580 \seq_clear:N \g_stex_modules_in_file_seq
1581 % \exp_args:Nnx \use:nn {
1582     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1583         \seq_clear:N \l_stex_all_modules_seq
1584         \prop_clear:N \l_stex_current_module_prop
1585         \str_set:Nx \l_tmpb_str { #2 }
1586         \str_if_empty:NF \l_tmpb_str {
1587             \stex_set_current_repository:n { #2 }
1588         }
1589         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1590         \input { \g__stex_importmodule_file_str }
1591     }
1592 % }{
1593
1594 % }
1595 \prop_gput:Noo \g_stex_module_files_prop
1596 \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1597 \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1598
1599 \stex_if_module_exists:nF { #1 ? #4 } {
1600     \msg_error:nnn{stex}{error/unknownmodule}{
1601         #1?#4~(in~file~\g__stex_importmodule_file_str)
1602     }
1603 }
1604 }
1605 \stex_activate_module:n { #1 ? #4 }
1606 }

```

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

`\importmodule`

```

1607 \NewDocumentCommand \importmodule { 0{} m } {
1608     \stex_import_module_uri:nn { #1 } { #2 }
1609     \stex_debug:nn{modules}{Importing~module:~
1610         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1611     }
1612     \stex_if_smsmode:F {
1613         \stex_import_require_module:nnnn
1614         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1615         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1616         \stex_annotate_invisible:nnn
1617         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}

```

```

1618 }
1619 \exp_args:Nx \stex_add_to_current_module:n {
1620   \stex_import_require_module:nnnn
1621   { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1622   { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1623 }
1624 \exp_args:Nx \stex_add_import_to_current_module:n {
1625   \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1626 }
1627 \stex_smsmode_set_codes:
1628 }
1629 \stex_deactivate_macro:Nn \importmodule {module~environments}

```

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

`\usemodule`

```

1630 \NewDocumentCommand \usemodule { 0{} m } {
1631   \stex_if_smsmode:F {
1632     \stex_import_module_uri:nn { #1 } { #2 }
1633     \stex_import_require_module:nnnn
1634     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1635     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1636     \stex_annotate_invisible:nnn
1637     {usemodule} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1638   }
1639   \stex_smsmode_set_codes:
1640 }

```

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

```

1641 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1642 ⟨*package⟩
1643
1644 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1645
```

Warnings and error messages

```
1646
```

23.1 Symbol Declarations

```
1647 ⟨@@=stex_symdecl⟩
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1648 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1649 \NewDocumentCommand \STEXsymbol { m } {
1650   \stex_get_symbol:n { #1 }
1651   \exp_args:No
1652   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1653 }
```

(End definition for \STEXsymbol. This function is documented on page 27.)

symdecl arguments:

```
1654 \keys_define:nn { stex / symdecl } {
1655   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1656   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1657   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1658   type      .tl_set:N    = \l_stex_symdecl_type_tl ,
1659   align     .str_set:N    = \l_stex_symdecl_align_str , % TODO(?)
1660   gfc       .str_set:N    = \l_stex_symdecl_gfc_str , % TODO(?)
1661   specializes .str_set:N  = \l_stex_symdecl_specializes_str , % TODO(?)
1662   def       .tl_set:N    = \l_stex_symdecl_definiens_tl
1663 }
```

```

1664
1665 \bool_new:N \l_stex_symdecl_make_macro_bool
1666
1667 \cs_new_protected:Nn \__stex_symdecl_args:n {
1668   \str_clear:N \l_stex_symdecl_name_str
1669   \str_clear:N \l_stex_symdecl_args_str
1670   \bool_set_false:N \l_stex_symdecl_local_bool
1671   \tl_clear:N \l_stex_symdecl_type_tl
1672   \tl_clear:N \l_stex_symdecl_definiens_tl
1673
1674   \keys_set:nn { stex / symdecl } { #1 }
1675 }

```

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

```

1676
1677 \NewDocumentCommand \symdecl { s O{} m } {
1678   \__stex_symdecl_args:n { #2 }
1679   \IfBooleanTF #1 {
1680     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1681   } {
1682     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1683   }
1684   \stex_symdecl_do:n { #3 }
1685   \stex_smsmode_set_codes:
1686 }
1687 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1688 \cs_new_protected:Nn \stex_symdecl_do:n {
1689   \stex_if_in_module:F {
1690     % TODO throw error? some default namespace?
1691   }
1692
1693   \str_if_empty:NT \l_stex_symdecl_name_str {
1694     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1695   }
1696
1697   \prop_if_exist:cT { g_stex_symdecl_
1698     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1699     \prop_item:Nn \l_stex_current_module_prop {name} ?
1700     \l_stex_symdecl_name_str
1701     _prop
1702   }{
1703     % TODO throw error (beware of circular dependencies)
1704   }
1705
1706   \prop_clear:N \l_tmpa_prop
1707   \prop_put:Nnx \l_tmpa_prop { module } {
1708     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1709     \prop_item:Nn \l_stex_current_module_prop {name}
1710   }

```

```

1711 \seq_clear:N \l_tmpa_seq
1712 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1713 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1714 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1715 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1716
1717 \exp_args:No \stex_add_constant_to_current_module:n {
1718   \l_stex_symdecl_name_str
1719 }
1720
1721 % arity/args
1722 \int_zero:N \l_tmpb_int
1723
1724 \bool_set_true:N \l_tmpa_bool
1725 \str_map_inline:Nn \l_stex_symdecl_args_str {
1726   \token_case_meaning:NnF ##1 {
1727     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1728     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1729     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1730     {\tl_to_str:n a} {
1731       \bool_set_false:N \l_tmpa_bool
1732       \int_incr:N \l_tmpb_int
1733     }
1734     {\tl_to_str:n B} {
1735       \bool_set_false:N \l_tmpa_bool
1736       \int_incr:N \l_tmpb_int
1737     }
1738   }{
1739     \msg_set:nnn{stex}{error/wrongargs}{
1740       args~value~in~symbol~declaration~for~
1741       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1742       \prop_item:Nn \l_stex_current_module_prop {name} ?
1743       \l_stex_symdecl_name_str ~
1744       needs~to~be~
1745       i,~a,~b~or~B,~but~##1~given
1746     }
1747     \msg_error:nn{stex}{error/wrongargs}
1748   }
1749 }
1750 \bool_if:NTF \l_tmpa_bool {
1751   % possibly numeric
1752   \str_if_empty:NTF \l_stex_symdecl_args_str {
1753     \prop_put:Nnn \l_tmpa_prop { args } {}
1754     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1755   }{
1756     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1757     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1758     \str_clear:N \l_tmpa_str
1759     \int_step_inline:nn \l_tmpa_int {
1760       \str_put_right:Nn \l_tmpa_str i
1761     }
1762     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1763   }
1764 } {

```

```

1765 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1766 \prop_put:Nnx \l_tmpa_prop { arity }
1767 { \str_count:N \l_stex_symdecl_args_str }
1768 }
1769 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1770
1771
1772 % semantic macro
1773
1774 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1775   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1776     \prop_item:Nn \l_tmpa_prop { module } ?
1777     \prop_item:Nn \l_tmpa_prop { name }
1778   } }
1779
1780   \bool_if:NF \l_stex_symdecl_local_bool {
1781     \exp_args:Nx \stex_add_to_current_module:n {
1782       \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1783         \prop_item:Nn \l_tmpa_prop { module } ?
1784         \prop_item:Nn \l_tmpa_prop { name }
1785       } }
1786     }
1787   }
1788 }
1789
1790 % add to all symbols
1791
1792 \bool_if:NF \l_stex_symdecl_local_bool {
1793   \exp_args:Nx \stex_add_to_current_module:n {
1794     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1795       \prop_item:Nn \l_tmpa_prop { module } ?
1796       \prop_item:Nn \l_tmpa_prop { name }
1797     }
1798   }
1799 }
1800
1801 \stex_debug:nn{symbols}{New~symbol:~
1802   \prop_item:Nn \l_tmpa_prop { module } ?
1803   \prop_item:Nn \l_tmpa_prop { name } ^^J
1804   Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1805   Args:~\prop_item:Nn \l_tmpa_prop { args }
1806 }
1807
1808 % circular dependencies require this:
1809
1810 \prop_if_exist:cF {
1811   g_stex_symdecl_
1812   \prop_item:Nn \l_tmpa_prop { module } ?
1813   \prop_item:Nn \l_tmpa_prop { name }
1814   _prop
1815 } {
1816   \prop_gset_eq:cN {
1817     g_stex_symdecl_
1818     \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1819     \prop_item:Nn \l_tmpa_prop { name }
1820     _prop
1821   } \l_tmpa_prop
1822 }
1823
1824 \stex_if_smsmode:TF {
1825   \bool_if:NF \l_stex_symdecl_local_bool {
1826     \exp_args:Nx \stex_add_to_sms:n {
1827       \prop_gset_from_keyval:cn {
1828         g_stex_symdecl_
1829         \prop_item:Nn \l_tmpa_prop { module } ?
1830         \prop_item:Nn \l_tmpa_prop { name }
1831         _prop
1832       } {
1833         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1834         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1835         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1836         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1837         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1838         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1839         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1840         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1841       }
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   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1850     \prop_item:Nn \l_tmpa_prop { module } ?
1851     \prop_item:Nn \l_tmpa_prop { name }
1852   }
1853   \stex_if_do_html:T {
1854     \stex_annotate_invisible:nnn {symdecl} {
1855       \prop_item:Nn \l_tmpa_prop { module } ?
1856       \prop_item:Nn \l_tmpa_prop { name }
1857     } {
1858       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1859       \stex_annotate_invisible:nnn{args}{}{
1860         \prop_item:Nn \l_tmpa_prop { args }
1861       }
1862       \stex_annotate_invisible:nnn{macroname}{}{#1}
1863       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1864         \stex_annotate_invisible:nnn{definiens}{}
1865         {\l_stex_symdecl_definiens_tl$}
1866       }
1867     }
1868   }
1869 }
1870 }

```

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

`\stex_get_symbol:n`

```
1871 \str_new:N \l_stex_get_symbol_uri_str
1872
1873 \cs_new_protected:Nn \stex_get_symbol:n {
1874   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1875     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1876   }{
1877     % argument is a string
1878     % is it a command name?
1879     \cs_if_exist:cTF { #1 }{
1880       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1881       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1882       \str_if_empty:NNTF \l_tmpa_str {
1883         \exp_args:Nx \cs_if_eq:NNTF {
1884           \tl_head:N \l_tmpa_tl
1885         } \stex_invoke_symbol:n {
1886           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1887         }{
1888           \__stex_symdecl_get_symbol_from_string:n { #1 }
1889         }
1890       } {
1891         \__stex_symdecl_get_symbol_from_string:n { #1 }
1892       }
1893     }{
1894       % argument is not a command name
1895       \__stex_symdecl_get_symbol_from_string:n { #1 }
1896       % \l_stex_all_symbols_seq
1897     }
1898   }
1899 }
1900
1901 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1902   \str_set:Nn \l_tmpa_str { #1 }
1903   \bool_set_false:N \l_tmpa_bool
1904   \stex_if_in_module:T {
1905     \prop_get:NnN \l_stex_current_module_prop
1906     { constants } \l_tmpa_seq
1907     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1908       \bool_set_true:N \l_tmpa_bool
1909       \str_set:Nx \l_stex_get_symbol_uri_str {
1910         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1911         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1912       }
1913     }
1914   }
1915   \bool_if:NF \l_tmpa_bool {
1916     \tl_set:Nn \l_tmpa_tl {
1917       \msg_set:nnn{stex}{error/unknownsymbol}{
1918         No~symbol~#1~found!
1919       }
1920     }
1921     \msg_error:nn{stex}{error/unknownsymbol}
1922   }
1923   \str_set:Nn \l_tmpa_str { #1 }
1924   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```



```

1924 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1925   \str_set:Nn \l_tmpb_str { ##1 }
1926   \str_if_eq:eeT { \l_tmpa_str } {
1927     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1928   } {
1929     \seq_map_break:n {
1930       \tl_set:Nn \l_tmpa_tl {
1931         \str_set:Nn \l_stex_get_symbol_uri_str {
1932           ##1
1933         }
1934       }
1935     }
1936   }
1937 }
1938 \l_tmpa_tl
1939 }
1940 }
1941
1942 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1943   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1944     { \tl_tail:N \l_tmpa_tl }
1945   \tl_if_single:NTF \l_tmpa_tl {
1946     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1947       \exp_after:wN \str_set:Nn \exp_after:wN
1948         \l_stex_get_symbol_uri_str \l_tmpa_tl
1949     }{
1950       % TODO
1951       % tail is not a single group
1952     }
1953   }{
1954     % TODO
1955     % tail is not a single group
1956   }
1957 }

```

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

23.2 Notations

```

1958 <@@=stex_notation>
1959 notation arguments:
1960 \keys_define:nn { stex / notation } {
1961   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1962   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1963   prec .str_set_x:N = \l__stex_notation_prec_str ,
1964   op .tl_set:N = \l__stex_notation_op_tl ,
1965   unknown .code:n = \str_set:Nx
1966     \l__stex_notation_variant_str \l_keys_key_str
1967 }
1968 \cs_new_protected:Nn \__stex_notation_args:n {
1969   \str_clear:N \l__stex_notation_lang_str
1970   \str_clear:N \l__stex_notation_variant_str

```

```

1971 \str_clear:N \l__stex_notation_prec_str
1972 \tl_clear:N \l__stex_notation_op_tl
1973
1974 \keys_set:nn { stex / notation } { #1 }
1975 }

```

\notation

```

1976 \NewDocumentCommand \notation { 0{ } m } {
1977   \__stex_notation_args:n { #1 }
1978   \tl_clear:N \l_stex_symdecl_definiens_tl
1979   \stex_get_symbol:n { #2 }
1980   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1981 }
1982 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

1983 \cs_new_protected:Nn \stex_notation_do:nn {
1984   \prop_set_eq:Nc \l_tmpa_prop {
1985     g_stex_symdecl_ #1 _prop
1986   }
1987
1988   \prop_clear:N \l_tmpb_prop
1989   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1990   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1991   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1992
1993   % precedences
1994   \seq_clear:N \l_tmpb_seq
1995   \exp_args:NNno
1996   \str_if_empty:NTF \l__stex_notation_prec_str {
1997     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1998     \int_compare:nNnTF \l_tmpa_str = 0 {
1999       \exp_args:NNnx
2000       \prop_put:Nno \l_tmpb_prop { opprec }
2001       { \neginfprec }
2002     }{
2003       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2004     }
2005   } {
2006     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2007       \exp_args:NNnx
2008       \prop_put:Nno \l_tmpb_prop { opprec }
2009       { \neginfprec }
2010       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2011       \int_step_inline:nn { \l_tmpa_str } {
2012         \exp_args:NNx
2013         \seq_put_right:Nn \l_tmpb_seq { \infprec }
2014       }
2015     }{
2016       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2017       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2018         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2019         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

2020         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2021         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
2022         \seq_map_inline:Nn \l_tmpa_seq {
2023             \seq_put_right:Nn \l_tmpb_seq { ##1 }
2024         }
2025     }
2026     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2027 }{
2028     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2029     \int_compare:nNnTF \l_tmpa_str = 0 {
2030         \exp_args:NNnx
2031         \prop_put:Nno \l_tmpb_prop { opprec }
2032         { \infprec }
2033     }{
2034         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2035     }
2036 }
2037 }
2038 }
2039
2040 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2041 \int_step_inline:nn { \l_tmpa_str } {
2042     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
2043         \exp_args:NNx
2044         \seq_put_right:Nn \l_tmpb_seq {
2045             \prop_item:Nn \l_tmpb_prop { opprec }
2046         }
2047     }
2048 }
2049
2050 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2051 \tl_clear:N \l_tmpa_tl
2052
2053 \int_compare:nNnTF \l_tmpa_str = 0 {
2054     \exp_args:NNe
2055     \cs_set:Npn \l__stex_notation_macrocode_cs {
2056         \_stex_term_math_oms:nnnn { #1 }
2057         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2058         { \prop_item:Nn \l_tmpb_prop { opprec } }
2059         { \exp_not:n { #2 } }
2060     }
2061     \__stex_notation_final:
2062 }{
2063     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2064     \str_if_in:NnTF \l_tmpb_str b {
2065         \exp_args:Nne \use:nn
2066         {
2067             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2068             \cs_set:Npn \l_tmpa_str { {
2069                 \_stex_term_math_omb:nnnn { #1 }
2070                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2071                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2072                 { \exp_not:n { #2 } }
2073             }}

```

```

2074   }{
2075     \str_if_in:NnTF \l_tmpb_str B {
2076       \exp_args:Nne \use:nn
2077       {
2078         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2079         \cs_set:Npn \l_tmpa_str } { {
2080           \stex_term_math_omb:nnnn { #1 }
2081           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2082           { \prop_item:Nn \l_tmpb_prop { opprec } }
2083           { \exp_not:n { #2 } }
2084         } }
2085       }{
2086         \exp_args:Nne \use:nn
2087         {
2088           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2089           \cs_set:Npn \l_tmpa_str } { {
2090             \stex_term_math_oma:nnnn { #1 }
2091             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2092             { \prop_item:Nn \l_tmpb_prop { opprec } }
2093             { \exp_not:n { #2 } }
2094           } }
2095         }
2096       }
2097
2098     \int_zero:N \l_tmpa_int
2099     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2100     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2101     \__stex_notation_arguments:
2102   }
2103 }

```

(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

```

2104 \cs_new_protected:Nn \__stex_notation_arguments: {
2105   \int_incr:N \l_tmpa_int
2106   \str_if_empty:NNTF \l_tmpa_str {
2107     \__stex_notation_final:
2108   }{
2109     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2110     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2111     \str_if_eq:VnTF \l_tmpb_str a {
2112       \__stex_notation_argument_assoc:n
2113     }{
2114       \str_if_eq:VnTF \l_tmpb_str B {
2115         \__stex_notation_argument_assoc:n
2116       }{
2117         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2118         \tl_put_right:Nx \l_tmpa_tl {
2119           { \stex_term_math_arg:nnn
2120             { \int_use:N \l_tmpa_int }
2121             { \l_tmpb_str }
2122             { ####\int_use:N \l_tmpa_int }
2123           }

```

```

2124     }
2125     \__stex_notation_arguments:
2126   }
2127 }
2128 }
2129 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2130 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2131   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2132   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2133   \tl_put_right:Nx \l_tmpa_tl {
2134     { \stex_term_math_assoc_arg:nnnn
2135       { \int_use:N \l_tmpa_int }
2136       { \l_tmpb_str }
2137       \exp_args:No \exp_not:n
2138       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2139       { ####\int_use:N \l_tmpa_int }
2140     }
2141   }
2142   \__stex_notation_arguments:
2143 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2144 \cs_new_protected:Nn \__stex_notation_final: {
2145   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2146   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2147   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2148   \exp_args:Nne \use:nn
2149   {
2150     \cs_generate_from_arg_count:cNnn {
2151       stex_notation_ \l_tmpa_str \c_hash_str
2152       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2153       _cs
2154     }
2155     \cs_gset:Npn \l_tmpb_str { { {
2156       \exp_after:wN \exp_after:wN \exp_after:wN
2157       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2158       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2159     } } }
2160
2161     \tl_if_empty:NF \l__stex_notation_op_tl {
2162       \cs_gset:cpx {
2163         stex_op_notation_ \l_tmpa_str \c_hash_str
2164         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2165         _cs
2166       } {
2167         \stex_term_oms:nnn {
2168           \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2169           \l__stex_notation_lang_str

```

```

2170     }{
2171         \l_tmpa_str
2172     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2173 }
2174 }
2175
2176
2177
2178 \stex_debug:nn{symbols}{
2179     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2180     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2181     Operator~precedence:~
2182     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2183     Argument~precedences:~
2184     \seq_use:Nn \l_tmpa_seq {,~}^^J
2185     Notation: \cs_meaning:c {
2186         stex_notation_ \l_tmpa_str \c_hash_str
2187         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2188         _cs
2189     }
2190 }
2191
2192 \prop_gset_eq:cN {
2193     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2194     \c_hash_str \l__stex_notation_lang_str _prop
2195 } \l_tmpb_prop
2196
2197 \exp_args:Nx
2198 \stex_add_to_current_module:n {
2199     \prop_get:cnN {
2200         g_stex_symdecl_
2201         \prop_item:Nn \l_tmpb_prop { symbol }
2202         _prop
2203     } { notations } \exp_not:N \l_tmpa_seq
2204     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2205         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2206     }
2207     \prop_put:cno {
2208         g_stex_symdecl_
2209         \prop_item:Nn \l_tmpb_prop { symbol }
2210         _prop
2211     } { notations } \exp_not:N \l_tmpa_seq
2212 }
2213
2214 \stex_if_smsmode:TF {
2215     \stex_smsmode_set_codes:
2216     \exp_args:Nx \stex_add_to_sms:n {
2217         \prop_gset_from_keyval:cn {
2218             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2219             \c_hash_str \l__stex_notation_lang_str _prop
2220         } {
2221             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2222             language = \prop_item:Nn \l_tmpb_prop { language } ,
2223             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2224         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2225         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2226     }
2227 }
2228 }{
2229   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2230   \seq_put_right:Nx \l_tmpa_seq {
2231     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2232   }
2233   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2234   \prop_set_eq:cN {
2235     g_stex_symdecl_ \l_tmpa_str _prop
2236   } \l_tmpa_prop
2237
2238   % HTML annotations
2239   \stex_if_do_html:T {
2240     \stex_annotate_invisible:nnn { notation }
2241     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2242       \stex_annotate_invisible:nnn { notationfragment }
2243       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2244       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2245       \stex_annotate_invisible:nnn { precedence }
2246       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2247         \seq_use:Nn \l_tmpa_seq { x }
2248       }{}
2249
2250       \int_zero:N \l_tmpa_int
2251       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2252       \tl_clear:N \l_tmpa_tl
2253       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2254         \int_incr:N \l_tmpa_int
2255         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2256         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2257         \str_if_eq:VnTF \l_tmpb_str a {
2258           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2259             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2260             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2261           } }
2262         }{
2263           \str_if_eq:VnTF \l_tmpb_str B {
2264             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2265               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2266               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2267             } }
2268           }{
2269             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2270               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2271             } }
2272           }
2273         }
2274       }
2275       \stex_annotate_invisible:nnn { notationcomp }{}{
2276         $ \exp_args:Nno \use:nn { \use:c {
2277           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2278         \c_hash_str \l__stex_notation_variant_str
2279         \c_hash_str \l__stex_notation_lang_str _cs
2280     } } { \l_tmpa_tl } $
2281   }
2282 }
2283 }
2284 }
2285 }

```

(End definition for _stex_notation_final:.)

\symdef

```

2286 \keys_define:nn { stex / symdef } {
2287   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2288   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2289   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2290   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2291   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2292   op        .tl_set:N   = \l__stex_notation_op_tl ,
2293   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2294   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2295   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2296   unknown   .code:n     = \str_set:Nx
2297             \l__stex_notation_variant_str \l_keys_key_str
2298 }
2299
2300 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2301   \str_clear:N \l_stex_symdecl_name_str
2302   \str_clear:N \l_stex_symdecl_args_str
2303   \bool_set_false:N \l_stex_symdecl_local_bool
2304   \tl_clear:N \l_stex_symdecl_type_tl
2305   \tl_clear:N \l_stex_symdecl_definiens_tl
2306   \str_clear:N \l__stex_notation_lang_str
2307   \str_clear:N \l__stex_notation_variant_str
2308   \str_clear:N \l__stex_notation_prec_str
2309   \tl_clear:N \l__stex_notation_op_tl
2310
2311   \keys_set:nn { stex / symdef } { #1 }
2312 }
2313
2314 \NewDocumentCommand \symdef { 0{} m } {
2315   \_stex_notation_symdef_args:n { #1 }
2316   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2317   \stex_symdecl_do:n { #2 }
2318   \exp_args:Nx \stex_notation_do:nn {
2319     \prop_item:Nn \l_tmpa_prop { module } ?
2320     \prop_item:Nn \l_tmpa_prop { name }
2321   }
2322 }
2323 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2324 \endpackage

```


Chapter 24

STEX -Terms Implementation

```
2325 <*package>
2326
2327 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2328
2329 <@@=stex_terms>
2330
2331 Warnings and error messages
2332 \msg_new:nnn{stex}{error/nonotation}{
2333   Symbol~#1~invoked,~but~has~no~notation~#2!
2334 }
2335 \msg_new:nnn{stex}{error/notationarg}{
2336   Error~in~parsing~notation~#1
2337 }
2338 \msg_new:nnn{stex}{error/noop}{
2339   Symbol~#1~has~no~operator~notation~for~notation~#2
2340 }
```

24.1 Symbol Invocations

Arguments:

```
2340 \keys_define:nn { stex / terms } {
2341   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2342   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2343   unknown .code:n = \str_set:Nx
2344     \l__stex_terms_variant_str \l_keys_key_str
2345 }
2346
2347 \cs_new_protected:Nn \__stex_terms_args:n {
2348   \str_clear:N \l__stex_terms_lang_str
2349   \str_clear:N \l__stex_terms_variant_str
2350   \str_clear:N \l__stex_terms_prec_str
2351   \tl_clear:N \l__stex_terms_op_tl
2352
2353   \keys_set:nn { stex / terms } { #1 }
```

2354 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2355 \cs_new_protected:Nn \stex_invoke_symbol:n {
2356   \if_mode_math:
2357     \exp_after:wN \__stex_terms_invoke_math:n
2358   \else:
2359     \exp_after:wN \__stex_terms_invoke_text:n
2360   \fi: { #1 }
2361 }
```

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

__stex_terms_invoke_math:n

```
2362 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2363   \peek_charcode_remove:NTF ! {
2364     \peek_charcode:NTF [ {
2365       \__stex_terms_invoke_op:nw { #1 }
2366     }{
2367       \peek_charcode_remove:NTF ! {
2368         \peek_charcode:NTF [ {
2369           \__stex_terms_invoke_op_custom:nw
2370         }{
2371           % TODO throw error
2372         }
2373       }{
2374         \__stex_terms_invoke_op:nw { #1 } []
2375       }
2376     }
2377   }{
2378     \peek_charcode_remove:NTF * {
2379       \__stex_terms_invoke_text:n { #1 }
2380     }{
2381       \peek_charcode:NTF [ {
2382         \__stex_terms_invoke_math:nw { #1 }
2383       }{
2384         \__stex_terms_invoke_math:nw { #1 } []
2385       }
2386     }
2387   }
2388 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2389 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2390   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2391     \stex_highlight_term:nn{#1}{#2}
2392   }
2393 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2394 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2395   \_stex_terms_args:n { #2 }
2396   \cs_if_exist:cTF {
2397     stex_op_notation_ #1 \c_hash_str
2398     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2399   }{
2400     \csname stex_op_notation_ #1 \c_hash_str
2401       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2402     \endcsname
2403   }{
2404     \msg_error:nnnn{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2405   }
2406 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2407 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2408   \_stex_terms_args:n { #2 }
2409   \prop_set_eq:Nc \l_tmpa_prop {
2410     g_stex_symdecl_ #1 _prop
2411   }
2412   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2413   \seq_if_empty:NTF \l_tmpa_seq {
2414     \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2415   } {
2416     \seq_if_in:NxTF \l_tmpa_seq
2417       { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2418       \use:c{
2419         stex_notation_ #1 \c_hash_str
2420         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2421         _cs
2422       }
2423     }{
2424       \str_if_empty:NTF \l__stex_terms_variant_str {
2425         \str_if_empty:NTF \l__stex_terms_lang_str {
2426           \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2427           \use:c{
2428             stex_notation_ #1 \c_hash_str \l_tmpa_str
2429             _cs
2430           }
2431         }{
2432           \msg_error:nn{stex}{error/nonotation}{#1}{
2433             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2434           }
2435         }
2436       }{
2437         \msg_error:nn{stex}{error/nonotation}{#1}{
2438           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2439         }
2440       }
2441     }
2442 }

```

```
2443 }
(End definition for \_stex_terms_invoke_math:nw.)
```

```
\_stex_terms_invoke_text:n
2444 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2445   \peek_charcode_remove:NTF ! {
2446     \stex_term_custom:nn { #1 } { }
2447   }{
2448     \prop_set_eq:Nc \l_tmpa_prop {
2449       g_stex_symdecl_ #1 _prop
2450     }
2451     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2452     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2453   }
2454 }
(End definition for \_stex_terms_invoke_text:n.)
```

24.2 Terms

Precedences:

```
\infprec
\neginfprec
\l__stex_terms_downprec
2455 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2456 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2457 \int_new:N \l__stex_terms_downprec
2458 \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
2459 \tl_set:Nn \l__stex_terms_left_bracket_str (
2460 \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

```
2461 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2462   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2463     \bool_set_false:N \l__stex_terms_brackets_done_bool
2464     #2
2465   } {
2466     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2467       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2468         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2469         \dobrackets { #2 }
2470       }
2471     }{ #2 }
2472   }
2473 }
```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```
2474 \bool_new:N \l__stex_terms_brackets_done_bool
2475 %\RequirePackage{scalerel}
2476 \cs_new_protected:Npn \dobrackets #1 {
2477   %\ThisStyle{\if D\m@switch
2478   %   \exp_args:Nnx \use:nn
2479   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2480   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2481   % \else
2482   \exp_args:Nnx \use:nn
2483   {
2484     \bool_set_true:N \l__stex_terms_brackets_done_bool
2485     \int_set:Nn \l__stex_terms_downprec \infpref
2486     \l__stex_terms_left_bracket_str
2487     #1
2488   }
2489   {
2490     \bool_set_false:N \l__stex_terms_brackets_done_bool
2491     \l__stex_terms_right_bracket_str
2492     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2493   }
2494   %\fi}
2495 }
```

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

`\withbrackets`

```
2496 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2497   \exp_args:Nnx \use:nn
2498   {
2499     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2500     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2501     #3
2502   }
2503   {
2504     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2505     {\l__stex_terms_left_bracket_str}
2506     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2507     {\l__stex_terms_right_bracket_str}
2508   }
2509 }
```

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

`\STEXinvisible`

```
2510 \cs_new_protected:Npn \STEXinvisible #1 {
2511   \stex_annotate_invisible:n { #1 }
2512 }
```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```
2513 \cs_new_protected:Nn \_stex_term_oms:nnn {
2514   \stex_annotate:nnn{ OMID }{ #2 }{
2515     \stex_highlight_term:nn { #1 } { #3 }
2516   }
2517 }
2518
2519 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2520   \__stex_terms_maybe_brackets:nn { #3 }{
2521     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2522   }
2523 }
```

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

`_stex_term_math_oma:nnnn`

```
2524 \cs_new_protected:Nn \_stex_term_oma:nnn {
2525   \stex_annotate:nnn{ OMA }{ #2 }{
2526     \stex_highlight_term:nn { #1 } { #3 }
2527   }
2528 }
2529
2530 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2531   \__stex_terms_maybe_brackets:nn { #3 }{
2532     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2533   }
2534 }
```

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

`_stex_term_math_omb:nnnn`

```
2535 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2536   \stex_annotate:nnn{ OMBIND }{ #2 }{
2537     \stex_highlight_term:nn { #1 } { #3 }
2538   }
2539 }
2540
2541 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2542   \__stex_terms_maybe_brackets:nn { #3 }{
2543     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2544   }
2545 }
```

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

`_stex_term_math_arg:nnn`

```
2546 \cs_new_protected:Nn \_stex_term_arg:nn {
2547   \stex_unhighlight_term:n {
2548     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2549   }
2550 }
2551 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2552   \exp_args:Nnx \use:nn
2553   { \int_set:Nn \l__stex_terms_downprec { #2 }

```

```

2554     \stex_term_arg:nn { #1 }{ #3 }
2555   }
2556   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2557 }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2558 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2559   \clist_set:Nn \l_tmpa_clist{ #4 }
2560   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2561     \tl_set:Nn \l_tmpa_tl { #4 }
2562   }{
2563     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2564     \clist_reverse:N \l_tmpa_clist
2565     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2566
2567     \clist_map_inline:Nn \l_tmpa_clist {
2568       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2569         \exp_args:Nno
2570         \l_tmpa_cs { ##1 } \l_tmpa_tl
2571       }
2572     }
2573
2574   }
2575   \exp_args:Nnno
2576   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2577 }

```

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

`\stex_term_custom:nn`

```

2578 \cs_new_protected:Nn \stex_term_custom:nn {
2579   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2580   \str_set:Nn \l_tmpa_str { #2 }
2581   \tl_clear:N \l_tmpa_tl
2582   \int_zero:N \l_tmpa_int
2583   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2584   \__stex_terms_custom_loop:
2585 }

```

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

`__stex_terms_custom_loop:`

```

2586 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2587   \bool_set_false:N \l_tmpa_bool
2588   \bool_while_do:nn {
2589     \str_if_eq_p:ee X {
2590       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2591     }
2592   }{
2593     \int_incr:N \l_tmpa_int
2594   }
2595
2596   \peek_charcode:NNTF [ {

```

```

2597 % notation/text component
2598 \__stex_terms_custom_component:w
2599 } {
2600 \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2601 % all arguments read => finish
2602 \__stex_terms_custom_final:
2603 } {
2604 % arguments missing
2605 \peek_charcode_remove:NTF * {
2606 % invisible, specific argument position or both
2607 \peek_charcode:NTF [ {
2608 % visible specific argument position
2609 \__stex_terms_custom_arg:wn
2610 } {
2611 % invisible
2612 \peek_charcode_remove:NTF * {
2613 % invisible specific argument position
2614 \__stex_terms_custom_arg_inv:wn
2615 } {
2616 % invisible next argument
2617 \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2618 }
2619 }
2620 } {
2621 % next normal argument
2622 \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2623 }
2624 }
2625 }
2626 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2627 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2628 \bool_set_true:N \l_tmpa_bool
2629 \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2630 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2631 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2632 \str_set:Nx \l_tmpb_str {
2633 \str_item:Nn \l_tmpa_str { #1 }
2634 }
2635 \str_case:VnTF \l_tmpb_str {
2636 { X } {
2637 \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2638 }
2639 { i } { \__stex_terms_custom_set_X:n { #1 } }
2640 { b } { \__stex_terms_custom_set_X:n { #1 } }
2641 { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2642 { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2643 }{}{

```



```

2644 \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2645 }
2646
2647 \bool_if:nTF \l_tmpa_bool {
2648   \tl_put_right:Nx \l_tmpa_tl {
2649     \stex_annotate_invisible:n {
2650       \stex_term_arg:nn { \int_eval:n { #1 } }
2651       \exp_not:n { { #2 } }
2652     }
2653   }
2654 } {
2655   \tl_put_right:Nx \l_tmpa_tl {
2656     \stex_term_arg:nn { \int_eval:n { #1 } }
2657     \exp_not:n { { #2 } }
2658   }
2659 }
2660
2661 \__stex_terms_custom_loop:
2662 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2663 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2664   \str_set:Nx \l_tmpa_str {
2665     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2666     X
2667     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2668   }
2669 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2670 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2671   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2672   \__stex_terms_custom_loop:
2673 }

```

(End definition for __stex_terms_custom_component:.)

__stex_terms_custom_final:

```

2674 \cs_new_protected:Nn \__stex_terms_custom_final: {
2675   \int_compare:nNnTF \l_tmpb_int = 0 {
2676     \exp_args:Nnno \stex_term_oms:nnn
2677   } {
2678     \str_if_in:NnTF \l_tmpa_str {b} {
2679       \exp_args:Nnno \stex_term_ombind:nnn
2680     } {
2681       \exp_args:Nnno \stex_term_oma:nnn
2682     }
2683   }
2684   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2685 }

```

(End definition for `_stex_terms_custom_final:.`)

```

\symref
\symname
2686 \NewDocumentCommand \symref { m m }{
2687   \let\compemph_uri_prev:\compemph@uri
2688   \let\compemph@uri\symrefemph@uri
2689   \STEXsymbol{#1}! [#2]
2690   \let\compemph@uri\compemph_uri_prev:
2691 }
2692
2693 \keys_define:nn { stex / symname } {
2694   post      .str_set_x:N   = \l_stex_symname_post_str
2695 }
2696
2697 \cs_new_protected:Nn \stex_symname_args:n {
2698   \str_clear:N \l_stex_symname_post_str
2699   \keys_set:nn { stex / symname } { #1 }
2700 }
2701
2702 \NewDocumentCommand \symname { 0{} m }{
2703   \stex_symname_args:n { #1 }
2704   \stex_get_symbol:n { #2 }
2705   \str_set:Nx \l_tmpa_str {
2706     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2707   }
2708   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2709
2710   \let\compemph_uri_prev:\compemph@uri
2711   \let\compemph@uri\symrefemph@uri
2712   \exp_args:NNx \use:nn
2713   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2714     \l_tmpa_str \l_stex_symname_post_str
2715   ] }
2716   \let\compemph@uri\compemph_uri_prev:
2717 }

```

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

24.3 Notation Components

2718 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2719
2720 \str_new:N \l__stex_notationcomps_highlight_uri_str
2721 \cs_new_protected:Nn \stex_highlight_term:nn {
2722   \exp_args:Nnx
2723   \use:nn {
2724     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2725     #2
2726   } {
2727     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2728     { \l__stex_notationcomps_highlight_uri_str }
2729   }

```

```

2730 }
2731
2732 \cs_new_protected:Nn \stex_unhighlight_term:n {
2733 % \latexml_if:TF {
2734 % #1
2735 % } {
2736 % \rustex_if:TF {
2737 % #1
2738 % } {
2739 #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2740 % }
2741 % }
2742 }

```

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

```

\comp
\compemph@uri 2743 \cs_new_protected:Npn \comp #1 {
\compemph 2744 \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph 2745 \rustex_if:TF {
\defemph@uri 2746 \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph 2747 }{
\symrefemph@uri 2748 \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2749 }
2750 }
2751 }
2752
2753 \cs_new_protected:Npn \compemph@uri #1 #2 {
2754 \compemph{ #1 }
2755 }
2756
2757
2758 \cs_new_protected:Npn \compemph #1 {
2759 \textcolor{blue}{#1}
2760 }
2761
2762 \cs_new_protected:Npn \defemph@uri #1 #2 {
2763 \defemph{#1}
2764 }
2765
2766 \cs_new_protected:Npn \defemph #1 {
2767 \textbf{#1}
2768 }
2769
2770 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2771 \symrefemph{#1}
2772 }
2773
2774 \cs_new_protected:Npn \symrefemph #1 {
2775 \textbf{#1}
2776 }

```

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

\ellipses

```
2777 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```
\parray
\prmatrix 2778 \bool_new:N \l_stex_inarray_bool
\parrayline 2779 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2780 \NewDocumentCommand \parray { m m } {
\parraycell 2781 \begingroup
2782 \bool_set_true:N \l_stex_inarray_bool
2783 \begin{array}{#1}
2784 #2
2785 \end{array}
2786 \endgroup
2787 }
2788
2789 \NewDocumentCommand \prmatrix { m } {
2790 \begingroup
2791 \bool_set_true:N \l_stex_inarray_bool
2792 \begin{matrix}
2793 #1
2794 \end{matrix}
2795 \endgroup
2796 }
2797
2798 \def \parrayline #1 #2 {
2799 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2800 }
2801
2802 \def \parraylineh #1 #2 {
2803 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2804 }
2805
2806 \def \parraycell #1 {
2807 #1 \bool_if:NT \l_stex_inarray_bool {&}
2808 }
```

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

```
2809 \endpackage
```

Chapter 25

STEX -Structural Features Implementation

```
2810 <*package>
2811
2812 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2813
2814 <@@=stex_features>
      Warnings and error messages
2815
```

25.1 The feature environment

structural@feature

```
2816
2817 \NewDocumentEnvironment{structural@feature}{ m m m }{
2818   \stex_if_in_module:F {
2819     \msg_set:nnn{stex}{error/nomodule}{
2820       Structural~Feature~has~to~occur~in~a~module:\\
2821       Feature~#2~of~type~#1\\
2822       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2823     }
2824     \msg_error:nn{stex}{error/nomodule}
2825   }
2826
2827   \str_set:Nx \l_stex_module_name_str {
2828     \prop_item:Nn \l_stex_current_module_prop
2829       { name } / #2 - feature
2830   }
2831
2832   \str_set:Nx \l_stex_module_ns_str {
2833     \prop_item:Nn \l_stex_current_module_prop
2834       { ns }
2835   }
2836
```

```

2837
2838 \str_clear:N \l_tmpa_str
2839 \seq_clear:N \l_tmpa_seq
2840 \tl_clear:N \l_tmpa_tl
2841 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2842   origname = #2,
2843   name     = \l_stex_module_name_str ,
2844   ns       = \l_stex_module_ns_str ,
2845   imports  = \exp_not:o { \l_tmpa_seq } ,
2846   constants = \exp_not:o { \l_tmpa_seq } ,
2847   content  = \exp_not:o { \l_tmpa_tl } ,
2848   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2849   lang     = \l_stex_module_lang_str ,
2850   sig      = \l_tmpa_str ,
2851   meta     = \l_tmpa_str ,
2852   feature  = #1 ,
2853 }
2854
2855 \stex_if_smsmode:TF {
2856   \stex_smsmode_set_codes:
2857 } {
2858   \begin{stex_annotate_env}{ feature:#1 }{}
2859   \stex_annotate_invisible:nnn{header}{}{ #3 }
2860 }
2861 }{
2862   \str_set:Nx \l_tmpa_str {
2863     c_stex_feature_
2864     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2865     \prop_item:Nn \l_stex_current_module_prop { name }
2866     _prop
2867   }
2868   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2869   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2870   \stex_if_smsmode:TF {
2871     \exp_args:Nx \stex_add_to_sms:n {
2872       \prop_gset_from_keyval:cn {
2873         c_stex_feature_
2874         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2875         \prop_item:Nn \l_stex_current_module_prop { name }
2876         _prop
2877       } {
2878         origname = #2,
2879         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2880         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2881         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2882         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2883         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2884         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2885         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2886         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2887         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2888         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2889       }
2890     }

```

```

2891 } {
2892     \end{stex_annotate_env}
2893 }
2894 }
2895

```

25.2 Features

structure

```

2896
2897 \prop_new:N \l_stex_all_structures_prop
2898
2899 \keys_define:nn { stex / features / structure } {
2900     name .str_set_x:N = \l__stex_features_structure_name_str ,
2901 }
2902
2903 \cs_new_protected:Nn \__stex_features_structure_args:n {
2904     \str_clear:N \l__stex_features_structure_name_str
2905     \keys_set:nn { stex / features / structure } { #1 }
2906 }
2907
2908 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2909 % \__stex_features_structure_args:n { ##1 }
2910 % \str_if_empty:NT \l__stex_features_structure_name_str {
2911 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2912 % }
2913 %} {
2914 %
2915 %}
2916
2917 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2918     \__stex_features_structure_args:n { #1 }
2919     \str_if_empty:NT \l__stex_features_structure_name_str {
2920         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2921     }
2922     \exp_args:Nnnx
2923     \begin{structural@feature}{ structure }
2924         { \l__stex_features_structure_name_str }{}
2925         \seq_clear:N \l_tmpa_seq
2926         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2927     }{
2928
2929         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2930         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2931         \str_set:Nx \l_tmpa_str {
2932             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2933             \prop_item:Nn \l_stex_current_module_prop { name }
2934         }
2935         \seq_map_inline:Nn \l_tmpa_seq {
2936             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2937         }
2938         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2939         \exp_args:Nnx

```

```

2940 \AddToHookNext { env / mathstructure / after }{
2941 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2942 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2943 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2944 \STEXexport {
2945 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2946 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2947 {\l_tmpa_str}
2948 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2949 {#2}{\l_tmpa_str}
2950 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2951 % \prop_item:Nn \l_stex_current_module_prop { origname },
2952 % \l_tmpa_str
2953 % }
2954 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2955 % #2,\l_tmpa_str
2956 % }
2957 % \tl_set:cx { #2 } {
2958 % \stex_invoke_structure:n { \l_tmpa_str }
2959 % }
2960 % }
2961
2962 \end{structural@feature}
2963 % \g_stex_last_feature_prop
2964 }

```

\instantiate

```

2965 \seq_new:N \l__stex_features_structure_field_seq
2966 \str_new:N \l__stex_features_structure_field_str
2967 \str_new:N \l__stex_features_structure_def_tl
2968 \prop_new:N \l__stex_features_structure_prop
2969 \NewDocumentCommand \instantiate { m O{} m }{
2970 \stex_smsmode_set_codes:
2971 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2972 \prop_set_eq:Nc \l__stex_features_structure_prop {
2973 c_stex_feature_\l_tmpa_str _prop
2974 }
2975 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2976 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2977 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2978 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2979 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2980 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2981 {!} \l_tmpa_tl
2982 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2983 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2984 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2985 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2986 }{
2987 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2988 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2989 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2990 \l_tmpa_tl
2991 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```



```

2992         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2993         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2994     }{
2995         \tl_clear:N \l_tmpb_tl
2996     }
2997 }
2998 }{
2999     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3000     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3001         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3002         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3003         \tl_clear:N \l_tmpa_tl
3004     }{
3005         % TODO throw error
3006     }
3007 }
3008 % \l_tmpa_str: name
3009 % \l_tmpa_tl: definiens
3010 % \l_tmpb_tl: notation
3011 \tl_if_empty:NT \l__stex_features_structure_field_str {
3012     % TODO throw error
3013 }
3014 \str_clear:N \l_tmpb_str
3015
3016 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3017 \seq_map_inline:Nn \l_tmpa_seq {
3018     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
3019     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3020     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3021         \seq_map_break:n {
3022             \str_set:Nn \l_tmpb_str { ####1 }
3023         }
3024     }
3025 }
3026 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
3027     \l_tmpb_str
3028
3029 \tl_if_empty:NTF \l_tmpb_tl {
3030     \tl_if_empty:NF \l_tmpa_tl {
3031         \exp_args:Nx \use:n {
3032             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3033         }
3034     }
3035 }{
3036     \tl_if_empty:NTF \l_tmpa_tl {
3037         \exp_args:Nx \use:n {
3038             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3039         }
3040     }
3041 }{
3042     \exp_args:Nx \use:n {
3043         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3044         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3045     }

```

```

3046     }
3047   }
3048   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3049   % \prop_item:Nn \l_stex_current_module_prop {name} ?
3050   % #3/\l_stex_features_structure_field_str
3051   % \par
3052   % \expandafter\present\csname
3053   %   g_stex_symdecl_
3054   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3055   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
3056   %   #3/\l_stex_features_structure_field_str
3057   %   _prop
3058   % \endcsname
3059 }
3060
3061 \tl_clear:N \l__stex_features_structure_def_tl
3062
3063 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3064 \seq_map_inline:Nn \l_tmpa_seq {
3065   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3066   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3067   \exp_args:Nx \use:n {
3068     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3069
3070     }
3071   }
3072
3073   \prop_if_exist:cF {
3074     g_stex_symdecl_
3075     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3076     \prop_item:Nn \l_stex_current_module_prop {name} ?
3077     #3/\l_tmpa_str
3078     _prop
3079   }{
3080     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
3081     \l_tmpb_str
3082     \exp_args:Nx \use:n {
3083       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3084     }
3085   }
3086 }
3087
3088 \symdecl*[type={\STEXsymbol{module-type}}{
3089   \_stex_term_math_oms:nnnn {
3090     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3091     \prop_item:Nn \l__stex_features_structure_prop {name}
3092     }{}{0}{}
3093   }{}{#3}
3094
3095 % TODO: -> sms file
3096
3097 \tl_set:cx{ #3 }{
3098   \stex_invoke_structure:nnn {
3099     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3100     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3101   } {
3102     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3103     \prop_item:Nn \l__stex_features_structure_prop {name}
3104   }
3105 }
3106
3107 }

```

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

\stex_invoke_structure:nnn

```

3108 % #1: URI of the instance
3109 % #2: URI of the instantiated module
3110 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3111   \tl_if_empty:nTF{ #3 }{
3112     \prop_set_eq:Nc \l__stex_features_structure_prop {
3113       c_stex_feature_ #2 _prop
3114     }
3115     \tl_clear:N \l_tmpa_tl
3116     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3117     \seq_map_inline:Nn \l_tmpa_seq {
3118       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3119       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3120       \cs_if_exist:cT {
3121         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3122       }{
3123         \tl_if_empty:NF \l_tmpa_tl {
3124           \tl_put_right:Nn \l_tmpa_tl {,}
3125         }
3126         \tl_put_right:Nx \l_tmpa_tl {
3127           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3128         }
3129       }
3130     }
3131     \exp_args:No \mathstrut \l_tmpa_tl
3132   }{
3133     \stex_invoke_symbol:n{#1/#3}
3134   }
3135 }

```

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

```

3136 </package>

```

Chapter 26

STEX -Statements Implementation

```
3137 <*package>
3138
3139 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3140
3141 \protected\def\ignorespacesandpars{
3142   \begingroup\catcode13=10\relax
3143   \@ifnextchar\par{
3144     \endgroup\expandafter\ignorespacesandpars\@gobble
3145   }{
3146     \endgroup
3147   }
3148 }
3149
3150 <@@=stex_statements>
3151
3152   Warnings and error messages
3153
3154 \def\titleemph#1{\textbf{#1}}
3155
symboldoc
3156 \NewDocumentEnvironment{symboldoc}{m}{
3157   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3158   \seq_clear:N \l_tmpb_seq
3159   \seq_map_inline:Nn \l_tmpa_seq {
3160     \str_if_eq:nnF{ ##1 }{}{
3161       \stex_get_symbol:n { ##1 }
3162       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3163         \l_stex_get_symbol_uri_str
3164       }
3165     }
3166   }
3167   \par
3168   \exp_args:Nnnx
3169   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3170 }
```

```

3168 \end{stex_annotate_env}
3169 }

3170 \seq_new:N \g_stex_statements_patched_seq
3171
3172 \cs_new_protected:Nn \stex_statements_set_patched:n {
3173   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3174 }
3175
3176 \cs_new_protected:Nn \stex_statements_patch:nn {
3177   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3178     \AddToHook{begindocument}{
3179       \cs_if_exist:cTF{end#1}{
3180         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3181         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3182       }{
3183         \NewDocumentEnvironment{#1}{0{}}{
3184           \use:c{__stex_statements_#2_begin:n}{ }
3185         }{
3186           \use:c{__stex_statements_#2_end:}
3187         }
3188       }
3189     }
3190   }
3191 }

```

26.1 Definitions

definition

```

3192 \keys_define:nn {stex / definiendum }{
3193   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3194   root      .str_set:N     = \l__stex_statements_definiendum_root_str
3195 }
3196 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3197   \str_clear:N \l__stex_statements_definiendum_root_str
3198   \tl_clear:N \l__stex_statements_definiendum_post_tl
3199   \keys_set:nn { stex / definiendum }{ #1 }
3200 }
3201 \NewDocumentCommand \definiendum { 0{ } m m } {
3202   \__stex_statements_definiendum_args:n { #1 }
3203   \stex_get_symbol:n { #2 }
3204   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3205   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3206     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3207       \tl_set:Nn \l_tmpa_tl { #3 }
3208     } {
3209       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3210       \tl_set:Nn \l_tmpa_tl {
3211         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3212       }
3213     }
3214   } {
3215     \tl_set:Nn \l_tmpa_tl { #3 }

```

```

3216 }
3217
3218 % TODO root
3219 \rustex_if:TF {
3220   \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3221 } {
3222   \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3223 }
3224 }
3225 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3226
3227 \NewDocumentCommand \definame { 0{} m } {
3228   \__stex_statements_definiendum_args:n { #1 }
3229   % TODO: root
3230   \stex_get_symbol:n { #2 }
3231   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3232   \str_set:Nx \l_tmpa_str {
3233     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3234   }
3235   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3236   \rustex_if:TF {
3237     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3238       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3239     }
3240   } {
3241     \defemph@uri {
3242       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3243     } { \l_stex_get_symbol_uri_str }
3244   }
3245 }
3246 \stex_deactivate_macro:Nn \definame {definition~environments}
3247
3248 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3249   \stex_reactivate_macro:N \definiendum
3250   \stex_reactivate_macro:N \definame
3251   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3252   \seq_clear:N \l_tmpb_seq
3253   \seq_map_inline:Nn \l_tmpa_seq {
3254     \str_if_eq:nnF{ ##1 }{}{
3255       \stex_get_symbol:n { ##1 }
3256       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3257         \l_stex_get_symbol_uri_str
3258       }
3259     }
3260   }
3261   \stex_smsmode_set_codes:
3262   \exp_args:Nnnx
3263   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3264 }
3265
3266 \cs_new_protected:Nn \__stex_statements_defi_end: {
3267   \end{stex_annotate_env}
3268 }

```

Hook:

```
3269 \stex_statements_patch:nn{definition}{defi}
      inline:
3270 \NewDocumentCommand \inlinedef { m } {
3271   \begingroup
3272   \stex_reactivate_macro:N \definiendum
3273   \stex_reactivate_macro:N \definame
3274   \stex_ref_new_doc_target:n{}
3275   #1
3276   \endgroup
3277 }
```

26.2 Assertions

assertion

```
3278 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3279   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3280   \seq_clear:N \l_tmpb_seq
3281   \seq_map_inline:Nn \l_tmpa_seq {
3282     \str_if_eq:nnF{ ##1 }{}{
3283       \stex_get_symbol:n { ##1 }
3284       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3285         \l_stex_get_symbol_uri_str
3286       }
3287     }
3288   }
3289   \titleemph{Assertion}~
3290   \stex_smsmode_set_codes:
3291   \exp_args:Nnnx
3292   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3293 }
3294
3295 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3296   \end{stex_annotate_env}
3297 }
```

Hook:

```
3298 \stex_statements_patch:nn{assertion}{assertion}
      inline:
3299 \NewDocumentCommand \inlineass { m } {
3300   \begingroup
3301   \stex_ref_new_doc_target:n{}
3302   #1
3303   \endgroup
3304 }
```

theorem

```
3305 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3306   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3307   \seq_clear:N \l_tmpb_seq
```

```

3308 \seq_map_inline:Nn \l_tmpa_seq {
3309   \str_if_eq:nnF{ ##1 }{}{
3310     \stex_get_symbol:n { ##1 }
3311     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3312       \l_stex_get_symbol_uri_str
3313     }
3314   }
3315 }
3316 \titleemph{Theorem}~
3317 \stex_smsmode_set_codes:
3318 \exp_args:Nnnx
3319 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3320 }
3321
3322 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3323   \end{stex_annotate_env}
3324 }

```

Hook:

```

3325 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3326 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3327   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3328   \seq_clear:N \l_tmpb_seq
3329   \seq_map_inline:Nn \l_tmpa_seq {
3330     \str_if_eq:nnF{ ##1 }{}{
3331       \stex_get_symbol:n { ##1 }
3332       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3333         \l_stex_get_symbol_uri_str
3334       }
3335     }
3336   }
3337   \titleemph{Lemma}~
3338   \stex_smsmode_set_codes:
3339   \exp_args:Nnnx
3340   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3341 }
3342
3343 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3344   \end{stex_annotate_env}
3345 }

```

Hook:

```

3346 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3347 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3348   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3349   \seq_clear:N \l_tmpb_seq
3350   \seq_map_inline:Nn \l_tmpa_seq {
3351     \str_if_eq:nnF{ ##1 }{}{
3352       \stex_get_symbol:n { ##1 }

```



```

3353     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3354       \l_stex_get_symbol_uri_str
3355     }
3356   }
3357 }
3358 \titleemph{Axiom}~
3359 \stex_smsmode_set_codes:
3360 \exp_args:Nnnx
3361 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3362 }
3363
3364 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3365   \end{stex_annotate_env}
3366 }

Hook:

3367 \stex_statements_patch:nn{axiom}{axiom}

```

26.3 Examples

example

```

3368 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3369   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3370   \seq_clear:N \l_tmpb_seq
3371   \seq_map_inline:Nn \l_tmpa_seq {
3372     \str_if_eq:nnF{ ##1 }{}{
3373       \stex_get_symbol:n { ##1 }
3374       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3375         \l_stex_get_symbol_uri_str
3376       }
3377     }
3378   }
3379   \titleemph{Example}~
3380   \stex_smsmode_set_codes:
3381   \exp_args:Nnnx
3382   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3383 }
3384
3385 \cs_new_protected:Nn \__stex_statements_example_end: {
3386   \end{stex_annotate_env}
3387 }

Hook:

3388 \stex_statements_patch:nn{example}{example}

inline:

3389 \NewDocumentCommand \inlineex { m } {
3390   \begingroup
3391   \stex_ref_new_doc_target:n{
3392     #1
3393   }
3394   \endgroup
3395 }

```

26.4 OMText

```

3395 \keys_define:nn { stex / omtext } {
3396   id      .str_set_x:N    = \l_stex_omtext_id_str ,
3397   title   .tl_set:N       = \l_stex_omtext_title_tl ,
3398   type    .tl_set_x:N     = \l_stex_omtext_type_tl ,
3399   for     .tl_set_x:N     = \l_stex_omtext_for_tl ,
3400   from    .tl_set_x:N     = \l_stex_omtext_from_tl ,
3401   start   .tl_set:N       = \l_stex_omtext_start_tl ,
3402 }
3403 \cs_new_protected:Nn \stex_omtext_args:n {
3404   \tl_clear:N \l_stex_omtext_title_tl
3405   \tl_clear:N \l_stex_omtext_start_tl
3406   \keys_set:nn { stex / omtext } { #1 }
3407 }
3408 \newif\if@in@omtext\@in@omtextfalse
3409 \NewDocumentEnvironment {omtext} { 0{ } } {
3410   \stex_omtext_args:n { #1 }
3411   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3412     \tl_if_empty:NF \l_stex_omtext_title_tl {
3413       \titleemph{\l_stex_omtext_title_tl}:~
3414     }
3415   }{
3416     \titleemph{\l_stex_omtext_start_tl}~
3417   }
3418   \@in@omtexttrue
3419
3420   \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3421   \stex_smsmode_set_codes:
3422   \ignorespacesandpars
3423 }{}
3424 \end{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

```
3425 <*package>
3426 <@@=stex_sproof>
3427
3428 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3429
```

27.2 Proofs

We first define some keys for the proof environment.

```
3430 \keys_define:nn { stex / spf } {
3431   id          .str_set:N = \l__stex_sproof_spf_id_str,
3432   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3433   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3434   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3435   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3436   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3437   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3438   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3439   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3440   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3441 }
3442 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3443   \str_clear:N \l__stex_sproof_spf_id_str
3444   \tl_clear:N \l__stex_sproof_spf_display_tl
3445   \tl_clear:N \l__stex_sproof_spf_for_tl
3446   \tl_clear:N \l__stex_sproof_spf_from_tl
3447   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3448   \tl_clear:N \l__stex_sproof_spf_type_tl
3449   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹⁰EDNOTE: need an implementation for L^AT_EX_ML

```

3450 \tl_clear:N \l__stex_sproof_spf_continues_tl
3451 \tl_clear:N \l__stex_sproof_spf_functions_tl
3452 \tl_clear:N \l__stex_sproof_spf_method_tl
3453 \keys_set:nn { stex / spf }{ #1 }
3454 }

```

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

```

3455 \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.

```

3456 \newcount\count_ten
3457 \newenvironment{pst@with@label}[1]{
3458   \edef\pst@label{#1}
3459   \advance\count_ten by 1\relax
3460   \count_ten=1
3461 }{
3462   \advance\count_ten by -1\relax
3463 }

```

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

```

3464 \def\the@pst@label{
3465   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3466 }

```

(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.

```

3467 \keys_define:nn { stex / pstlabel }{
3468   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3469   delimiter    .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3470   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3471 }
3472 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

3473 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3474 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3475 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3476 }
3477 \__stex_sproof_pstlabel_args:n {}
3478 \newcommand\setpstlabelstyle[1]{
3479   \__stex_sproof_pstlabel_args:n {#1}
3480 }
3481 \newcommand\setpstlabelstyledefault{%
3482   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3483 }

```

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

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

```

3484 \ExplSyntaxOff
3485 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3486 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3487 \def\pst@make@label@short#1#2{#2}
3488 \def\pst@make@label@empty#1#2{}
3489 \ExplSyntaxOn
3490 \def\pstlabelstyle#1{%
3491   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3492 }%
3493 \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.

```

3494 \def\next@pst@label{%
3495   \global\advance\count\count10 by 1%
3496 }%

```

(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

```

3497 \def\sproof@box{
3498   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3499 }
3500 \def\spf@proofend{\sproof@box}
3501 \def\sproofend{
3502   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3503     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3504   }
3505 }
3506 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3507 \def\spf@proofsketch@kw{Proof Sketch}
3508 \def\spf@proof@kw{Proof}
3509 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3510 \cs_if_exist:NT \bbl@loaded {
3511   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3512   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3513     \input{proof-ngerman.lda}
3514   }
3515   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3516     \input{proof-finnish.lda}
3517   }
3518   \clist_if_in:NnT \l_tmpa_clist {french}{
3519     \input{proof-french.lda}
3520   }
3521   \clist_if_in:NnT \l_tmpa_clist {russian}{
3522     \input{proof-russian.lda}
3523   }
3524 }
3525

```

spfsketch

```

3526 \newcommand\spfsketch[2] [] {
3527   \__stex_sproof_spf_args:n{#1}
3528   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3529     \titleemph{
3530       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3531         \spf@proofsketch@kw
3532       }{
3533         \l__stex_sproof_spf_type_tl
3534       }
3535     }:
3536   }
3537   {-#2}
3538   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3539   \sproofend
3540 }

```

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

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

```

3541 \newenvironment{spfeq}[2] [] {
3542   \__stex_sproof_spf_args:n{#1}
3543   %\sref@target
3544   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3545     \titleemph{
3546       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3547         \spf@proof@kw
3548       }{
3549         \l__stex_sproof_spf_type_tl
3550       }
3551     }:

```

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

¹²EDNOTE: document above

```

3552 }
3553 {~#2}
3554 \begin{displaymath}\begin{array}{rcll}
3555 }{
3556 \end{array}\end{displaymath}
3557 }

```

(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.

```

3558 \newenvironment{spf@proof}[2][]{
3559   \__stex_sproof_spf_args:n{#1}
3560   %\sref@target
3561   \count_ten=10
3562   \par\noindent
3563   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3564     \titleemph{
3565       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3566         \spf@proof@kw
3567       }{
3568         \l__stex_sproof_spf_type_tl
3569       }
3570     }:
3571   }
3572   {~#2}
3573   %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3574   \def\pst@label{}
3575   \newcount\pst@count% initialize the labeling mechanism
3576   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3577   }{
3578     \end{pst@with@label}\end{description}
3579   }
3580 \newenvironment{sproof}[2][{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3581 \newenvironment{sProof}[2][{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3582 \newcommand\spfidea[2][]{
3583   \__stex_sproof_spf_args:n{#1}
3584   \titleemph{
3585     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3586       \l__stex_sproof_spf_type_tl
3587     }:
3588   }~#2
3589   \sproofend
3590 }

```

(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

```

3591 \newenvironment{spfstep}[1][]{
3592   \_stex_sproof_spf_args:n{#1}
3593   \@in@omtexttrue
3594   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3595     \item[\the@pst@label]
3596   }
3597   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3598     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3599   }
3600   %\sref@label@id{\pst@label}
3601   \ignorespacesandpars
3602 }{
3603   \next@pst@label\ignorespacesandpars
3604 }

```

sproofcomment

```

3605 \newenvironment{sproofcomment}[1][]{
3606   \_stex_sproof_spf_args:n{#1}
3607   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3608     \item[\the@pst@label]
3609   }
3610 }{
3611   \next@pst@label
3612 }

```

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.

```

3613 \newenvironment{subproof}[2][]{
3614   \_stex_sproof_spf_args:n{#1}
3615   \def\@test{#2}
3616   \ifx\@test\empty\else
3617     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3618       \item[\the@pst@label]
3619     }{#2}
3620   \fi
3621   \begin{pst@with@label}{\pst@label,\number\count_ten}
3622 }{
3623   \end{pst@with@label}\next@pst@label
3624 }

```

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

```

3625 \newenvironment{spfcases}[2][]{
3626   \def\@test{#1}
3627   \ifx\@test\empty
3628     \begin{subproof}[method=by-cases]{#2}
3629   \else
3630     \begin{subproof}[#1,method=by-cases]{#2}
3631   \fi
3632 }{

```

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


```

3633 \end{subproof}
3634 }

```

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

```

3635 \newenvironment{spfcase}[2] [] {
3636   \__stex_sproof_spf_args:n{#1}
3637   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3638     \item[\the@pst@label]
3639   }
3640   \def\@test{#2}
3641   \ifx\@test\@empty
3642   \else
3643     {\titleemph{#2}:~}
3644   \fi
3645   \begin{pst@with@label}{\pst@label,\number\count_ten}
3646   }{
3647     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3648       \sproofend
3649     }
3650     \end{pst@with@label}
3651     \next@pst@label
3652   }

```

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

```

3653 \newcommand\spfcasesketch[3] [] {
3654   \__stex_sproof_spf_args:n{#1}
3655   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3656     \item[\the@pst@label]
3657   }
3658   \def\@test{#2}
3659   \ifx\@test\@empty
3660   \else
3661     {\titleemph{#2}:~}
3662   \fi#3
3663   \next@pst@label
3664 }%

```

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.

```

3665 \keys_define:nn { stex / just }{
3666   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3667   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3668   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3669   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3670 }

```

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

```
3671 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3672 \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.

```
3673 \newcommand\justarg[2] [] {#2}
```

```
3674 \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

```
3675 <*package>
3676
3677 %%%%%%%%%% others.dtx %%%%%%%%%%
3678
3679 <@@=stex_others>
    Warnings and error messages
3680 % None

\MSC Math subject classifier

3681 \NewDocumentCommand \MSC {m} {
3682   % TODO
3683 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3684 \@ifpackageloaded{tikzinput}{
3685   \RequirePackage{stex-tikzinput}
3686 }{}
3687 </package>
```

Chapter 29

STEX -Metatheory Implementation

```
3688 <*package>
3689 <@@=stex_modules>
3690
3691 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3692
3693 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3694 \begingroup
3695 \stex_module_setup:nn{
3696   ns=\c_stex_metatheory_ns_str,
3697   meta=NONE
3698 }{Metatheory}
3699 \stex_reactivate_macro:N \symdecl
3700 \stex_reactivate_macro:N \notation
3701 \stex_reactivate_macro:N \symdef
3702 \ExplSyntaxOff
3703 \csname stex_suppress_html:n\endcsname{
3704   % is-a (a:A, a \in A, a is an A, etc.)
3705   \symdecl[args=ai]{isa}
3706   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3707   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3708   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3709
3710   % bind (\forall, \Pi, \lambda etc.)
3711   \symdecl[args=Bi]{bind}
3712   \notation[forall]{bind}{\comp\forall #1. #2}{#1 \comp, #2}
3713   \notation[\Pi]{bind}{\comp\prod_{#1} #2}{#1 \comp, #2}
3714   \notation[deffun]{bind}{\comp( #1 \comp)\;\to\;}{#1 \comp, #2}
3715
3716   % dummy variable
3717   \symdecl{dummyvar}
3718   \notation[underscore]{dummyvar}{\comp\_}
3719   \notation[dot]{dummyvar}{\comp\cdot}
3720   \notation[dash]{dummyvar}{\comp{\rm --}}
3721
3722   %fromto (function space, Hom-set, implication etc.)
```

```

3723 \symdecl[args=ai]{fromto}
3724 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3725 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3726
3727 % mapto (lambda etc.)
3728 %\symdecl[args=Bi]{mapto}
3729 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3730 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3731 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3732
3733 % function/operator application
3734 \symdecl[args=ia]{apply}
3735 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3736 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3737
3738 % ‘‘type’’ of all collections (sets,classes,types,kinds)
3739 \symdecl{collection}
3740 \notation[U]{collection}{\comp{\mathcal{U}}}
3741 \notation[set]{collection}{\comp{\textsf{Set}}}
3742
3743 % sequences
3744 \symdecl[args=1]{seqtype}
3745 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3746
3747 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3748 \notation[ui]{sequence-index}{#1^{#2}}
3749
3750 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
3751 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
3752 % ^ superceded by \aseqfromto and \livar/\uivar
3753
3754 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3755 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses},#2}{#1\comp,#2}
3756 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
3757
3758 % letin (‘‘let’’, local definitions, variable substitution)
3759 \symdecl[args=bii]{letin}
3760 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3761 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3762 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3763
3764 % structures
3765 \symdecl*[args=1]{module-type}
3766 \notation{module-type}{\mathtt{MOD} #1}
3767 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3768 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3769
3770 }
3771 \ExplSyntaxOn
3772 \stex_add_to_current_module:n{
3773   \let\nappa\apply
3774   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3775   \def\livar{\csname sequence-index\endcsname[li]}
3776   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3777     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3778     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3779     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3780   }
3781   \__stex_modules_end_module:
3782   \endgroup
3783 \endpackage

```

Chapter 30

Tikzinput Implementation

```
3784 <*package>
3785
3786 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3787
3788 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3789 \RequirePackage{l3keys2e}
3790
3791 \keys_define:nn { tikzinput } {
3792   image .bool_set:N = \c_tikzinput_image_bool,
3793   image .default:n = false ,
3794   unknown .code:n = {}
3795 }
3796
3797 \ProcessKeysOptions { tikzinput }
3798
3799 \bool_if:NTF \c_tikzinput_image_bool {
3800   \RequirePackage{graphicx}
3801
3802   \providecommand\usetikzlibrary[]{}
3803   \newcommand\tikzinput[2] []{\includegraphics[#1]{#2}}
3804 }{
3805   \RequirePackage{tikz}
3806   \RequirePackage{standalone}
3807
3808   \newcommand \tikzinput [2] [] {
3809     \setkeys{Gin}{#1}
3810     \ifx \Gin@ewidth \Gin@exclamation
3811       \ifx \Gin@eheight \Gin@exclamation
3812         \input { #2 }
3813       \else
3814         \resizebox{!}{ \Gin@eheight }{
3815           \input { #2 }
3816         }
3817       \fi
3818     \else
3819       \ifx \Gin@eheight \Gin@exclamation
3820         \resizebox{ \Gin@ewidth }{!}{
3821           \input { #2 }
```

```

3822     }
3823     \else
3824         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3825             \input { #2 }
3826         }
3827     \fi
3828 \fi
3829 }
3830 }
3831
3832 \newcommand \ctikzinput [2] [] {
3833     \begin{center}
3834         \tikzinput [1] {#2}
3835     \end{center}
3836 }
3837
3838 \@ifpackageloaded{stex}{
3839     \RequirePackage{stex-tikzinput}
3840 }{}
3841
3842 </package>
3843 <*stex>
3844 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3845 \RequirePackage{stex}
3846 \RequirePackage{tikzinput}
3847
3848 \newcommand\mhtikzinput [2] [] {%
3849     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3850     \stex_in_repository:nn\Gin@mhrepos{
3851         \tikzinput [1]{\mhpath{##1}{#2}}
3852     }
3853 }
3854 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
3855 </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.

```
3856 \*cls)
3857 \@@=document_structure)
3858 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3859 \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`

```
3860 \keys_define:nn{ document-structure / pkg }{
3861   class      .str_set_x:N = \c_document_structure_class_str,
3862   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3863   report     .code:n      = {
3864     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3865     \str_set:Nn \c_document_structure_class_str {report}
3866   },
3867   book       .code:n      = {
3868     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3869     \str_set:Nn \c_document_structure_class_str {book}
3870   },
3871   bookpart   .code:n      = {
3872     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3873     \str_set:Nn \c_document_structure_class_str {book}
3874     \str_set:Nn \c_document_structure_topsect_str {chapter}
3875   },
```

```

3876 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3877 unknown     .code:n      = {
3878   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3879 }
3880 }
3881 \ProcessKeysOptions{ document-structure / pkg }
3882 \str_if_empty:NT \c_document_structure_class_str {
3883   \str_set:Nn \c_document_structure_class_str {article}
3884 }
3885 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3886   {\c_document_structure_class_str}
3887

```

31.3 Beefing up the document environment

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

```

3888 \RequirePackage{omdoc}
3889 \bool_if:NF \c_document_structure_minimal_bool {
3890   \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

```

3891 \keys_define:nn { document-structure / document }{
3892   id .str_set_x:N = \c_document_structure_document_id_str
3893 }
3894 \let\__document_structure_orig_document=\document
3895 \renewcommand{\document}[1][]{
3896   \keys_set:nn{ document-structure / document }{ #1 }
3897   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3898   \__document_structure_orig_document
3899 }

```

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

```

3900 }
3901 \</cls>

```

31.4 Implementation: OMDoc Package

```

3902 \*package>
3903 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3904 \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

```

3905
3906 \keys_define:nn{ document-structure / pkg }{
3907   class      .str_set_x:N = \c_document_structure_class_str,
3908   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3909   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3910 }
3911 \ProcessKeysOptions{ document-structure / pkg }
3912 \str_if_empty:NT \c_document_structure_class_str {
3913   \str_set:Nn \c_document_structure_class_str {article}
3914 }
3915 \str_if_empty:NT \c_document_structure_topsect_str {
3916   \str_set:Nn \c_document_structure_topsect_str {section}
3917 }

```

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

```

3918 \RequirePackage{xspace}
3919 \RequirePackage{comment}
3920 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3921 \@ifpackageloaded{babel}{
3922   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3923   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3924     \input{omdoc-ngerman.ldf}
3925   }
3926 }{}
3927 %\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.

```

3928 \int_new:N \l_document_structure_section_level_int
3929 \str_case:VnF \c_document_structure_topsect_str {
3930   {part}{
3931     \int_set:Nn \l_document_structure_section_level_int {0}
3932   }
3933   {chapter}{
3934     \int_set:Nn \l_document_structure_section_level_int {1}
3935   }
3936 }{
3937   \str_case:VnF \c_document_structure_class_str {
3938     {book}{
3939       \int_set:Nn \l_document_structure_section_level_int {0}
3940     }
3941     {report}{
3942       \int_set:Nn \l_document_structure_section_level_int {0}
3943     }
3944   }{
3945     \int_set:Nn \l_document_structure_section_level_int {2}
3946   }
3947 }

```

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

```
3948 \def\current@section@level{document}%
3949 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3950 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3951 \cs_new_protected:Npn \skipomgroup {
3952   \ifcase\l_document_structure_section_level_int
3953   \or\stepcounter{part}
3954   \or\stepcounter{chapter}
3955   \or\stepcounter{section}
3956   \or\stepcounter{subsection}
3957   \or\stepcounter{subsubsection}
3958   \or\stepcounter{paragraph}
3959   \or\stepcounter{subparagraph}
3960   \fi
3961 }
```

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

`blindomgroup`

```
3962 \newcommand\at@begin@blindomgroup[1]{%
3963 \newenvironment{blindomgroup}
3964 {
3965   \int_incr:N\l_document_structure_section_level_int
3966   \at@begin@blindomgroup\l_document_structure_section_level_int
3967 }{}}
```

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

```
3968 \newcommand\omgroup@nonum[2]{
3969   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3970   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3971 }
```

(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.

```
3972 \newcommand\omgroup@num[2]{
```

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

```

3973 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3974   \@nameuse{#1}{#2}
3975 }{
3976   \cs_if_exist:NTF\rdfmata@sectioning{
3977     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3978   }{
3979     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3980   }
3981 }
3982 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3983 }

```

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

omgroup

```

3984 \keys_define:nn { document-structure / omgroup }{
3985   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3986   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
3987   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
3988   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3989   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
3990   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
3991   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
3992   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
3993   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
3994   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
3995 }
3996 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
3997   \str_clear:N \l__document_structure_omgroup_id_str
3998   \str_clear:N \l__document_structure_omgroup_date_str
3999   \clist_clear:N \l__document_structure_omgroup_creators_clist
4000   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4001   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4002   \tl_clear:N \l__document_structure_omgroup_type_tl
4003   \tl_clear:N \l__document_structure_omgroup_short_tl
4004   \tl_clear:N \l__document_structure_omgroup_display_tl
4005   \tl_clear:N \l__document_structure_omgroup_intro_tl
4006   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4007   \keys_set:nn { document-structure / omgroup } { #1 }
4008 }

```

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.

```

4009 \newif\if@mainmatter\@mainmattertrue
4010 \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.

```

4011 \keys_define:nn { document-structure / sectioning }{
4012   name .str_set_x:N = \l__document_structure_sect_name_str ,
4013   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4014   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4015   num .bool_set:N = \l__document_structure_sect_num_bool ,
4016 }

```

```

4017 \cs_new_protected:Nn \__document_structure_sect_args:n {
4018   \str_clear:N \l__document_structure_sect_name_str
4019   \str_clear:N \l__document_structure_sect_ref_str
4020   \bool_set_false:N \l__document_structure_sect_clear_bool
4021   \bool_set_false:N \l__document_structure_sect_num_bool
4022   \keys_set:nn { document-structure / sectioning } { #1 }
4023 }
4024 \newcommand\omdoc@sectioning[3][]{
4025   \__document_structure_sect_args:n {#1}
4026   \let\omdoc@sect@name\l__document_structure_sect_name_str
4027   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4028   \if@mainmatter% numbering not overridden by frontmatter, etc.
4029     \bool_if:NTF \l__document_structure_sect_num_bool {
4030       \omgroup@num{#2}{#3}
4031     }{
4032       \omgroup@nonum{#2}{#3}
4033     }
4034     \def\current@section@level{\omdoc@sect@name}
4035   \else
4036     \omgroup@nonum{#2}{#3}
4037   \fi
4038 }% 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.

```

4039 \newcommand\omgroup@redefine@addtocontents[1]{%
4040   %\edef\__document_structureimport{#1}%
4041   %\@for\@I:=\__document_structureimport\do{%
4042     %\edef\@path{\csname module@\@I @path\endcsname}%
4043     %\@ifundefined{tf@toc}\relax%
4044     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4045   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4046   %\def\addcontentsline##1##2##3{%
4047     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4048   %\else% hyperref.sty not loaded
4049   %\def\addcontentsline##1##2##3{%
4050     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4051   %\fi
4052 }% 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.

```

4053 \int_new:N \l_document_structure_omgroup_level_int
4054 \newenvironment{omgroup}[2][]{% keys, title
4055 {
4056   \__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.

```

4057 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4058   \omgroup@redefine@addtocontents{
4059     %\@ifundefined{module@id}\used@modules%
4060     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4061     }
4062 }

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

4063 \int_incr:N \l_document_structure_omgroup_level_int
4064 \int_incr:N \l_document_structure_section_level_int
4065 \ifcase\l_document_structure_section_level_int
4066   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4067   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4068   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4069   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4070   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4071   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4072   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4073 \fi
4074 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4075 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
4076 }% for customization
4077 {}

```

and finally, we localize the sections

```

4078 \newcommand\omdoc@part@kw{Part}
4079 \newcommand\omdoc@chapter@kw{Chapter}
4080 \newcommand\omdoc@section@kw{Section}
4081 \newcommand\omdoc@subsection@kw{Subsection}
4082 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4083 \newcommand\omdoc@paragraph@kw{paragraph}
4084 \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`

```

4085 \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).

```

4086 \cs_if_exist:NTF\frontmatter{
4087   \let\__document_structure_orig_frontmatter\frontmatter
4088   \let\frontmatter\relax
4089 }{
4090   \tl_set:Nn\__document_structure_orig_frontmatter{
4091     \clearpage
4092     \@mainmatterfalse
4093     \pagenumbering{roman}
4094   }
4095 }
4096 \cs_if_exist:NTF\backmatter{

```

```

4097 \let\__document_structure_orig_backmatter\backmatter
4098 \let\backmatter\relax
4099 }{
4100 \tl_set:Nn\__document_structure_orig_backmatter{
4101 \clearpage
4102 \@mainmatterfalse
4103 \pagenumbering{roman}
4104 }
4105 }

```

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.

```

4106 \newenvironment{frontmatter}{
4107 \__document_structure_orig_frontmatter
4108 }{
4109 \cs_if_exist:NTF\mainmatter{
4110 \mainmatter
4111 }{
4112 \clearpage
4113 \@mainmattertrue
4114 \pagenumbering{arabic}
4115 }
4116 }

```

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

```

4117 \newenvironment{backmatter}{
4118 \__document_structure_orig_backmatter
4119 }{
4120 \cs_if_exist:NTF\mainmatter{
4121 \mainmatter
4122 }{
4123 \clearpage
4124 \@mainmattertrue
4125 \pagenumbering{arabic}
4126 }
4127 }

```

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

```

4128 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4129 \newcommand\afterprematurestop{}
4130 \def\prematurestop@endomgroup{
4131 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4132 \end{omgroup}
4133 \int_decr:N \l_document_structure_omgroup_level_int
4134 \prematurestop@endomgroup
4135 }
4136 }
4137 \providecommand\prematurestop{

```



```

4138 \message{Stopping sTeX processing prematurely}
4139 \prematurestop@endomgroup
4140 \afterprematurestop
4141 \end{document}
4142 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

4143 \RequirePackage{etoolbox}
4144 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4145 \newrobustcmd\useSGvar[1]{%
4146 \@ifundefined{sTeX@Gvar@#1}
4147 {\PackageError{omdoc}
4148 {The sTeX Global variable #1 is undefined}
4149 {set it with \protect\setSGvar}}
4150 \@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.

```

4151 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4152 \@ifundefined{sTeX@Gvar@#1}
4153 {\PackageError{omdoc}
4154 {The sTeX Global variable #1 is undefined}
4155 {set it with \protect\setSGvar}}
4156 {\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.

```
4157 \*cls)
4158 \@@=mikoslides)
4159 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4160 \RequirePackage{13keys2e,expl-keystr-compatible}
4161
4162 \keys_define:nn{mikoslides / cls}{
4163   class .code:n = {
4164     \PassOptionsToClass{\CurrentOption}{omdoc}
4165     \str_if_eq:nnT{#1}{book}{
4166       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4167     }
4168     \str_if_eq:nnT{#1}{report}{
4169       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4170     }
4171   },
4172   notes .bool_set:N = \c__mikoslides_notes_bool ,
4173   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4174   unknown .code:n = {
4175     \PassOptionsToClass{\CurrentOption}{omdoc}
4176     \PassOptionsToClass{\CurrentOption}{beamer}
4177     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4178   }
4179 }
4180 \ProcessKeysOptions{ mikoslides / cls }
4181 \bool_if:NTF \c__mikoslides_notes_bool {
4182   \PassOptionsToPackage{notes=true}{mikoslides}
4183 }{
4184   \PassOptionsToPackage{notes=false}{mikoslides}
4185 }
4186 \</cls)
```

now we do the same for the mikoslides package.

```

4187 <*package>
4188 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4189 \RequirePackage{l3keys2e,expl-keystr-compat}
4190
4191 \keys_define:nn{mikoslides / pkg}{
4192   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
4193   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4194   notes            .bool_set:N = \c__mikoslides_notes_bool ,
4195   slides           .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4196   sectocframes     .bool_set:N = \c__mikoslides_sectocframes_bool ,
4197   frameimages      .bool_set:N = \c__mikoslides_frameimages_bool ,
4198   fiboxed          .bool_set:N = \c__mikoslides_fiboxed_bool ,
4199   noproblems       .bool_set:N = \c__mikoslides_noproblems_bool,
4200   unknown          .code:n      = {
4201     \PassOptionsToClass{\CurrentOption}{stex}
4202     \PassOptionsToClass{\CurrentOption}{tikzinput}
4203   }
4204 }
4205 \ProcessKeysOptions{ mikoslides / pkg }
4206 \newif\ifnotes
4207 \bool_if:NTF \c__mikoslides_notes_bool {
4208   \notesttrue
4209 }{
4210   \notesfalse
4211 }
4212

```

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

```

4213 \str_if_empty:NTF \c__mikoslides_topsect_str {
4214   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4215 }{
4216   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4217 }
4218 </package>

```

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

```

4219 <*cls>
4220 \bool_if:NTF \c__mikoslides_notes_bool {
4221   \LoadClass{omdoc}
4222 }{
4223   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4224   \newcounter{Item}
4225   \newcounter{paragraph}
4226   \newcounter{subparagraph}
4227   \newcounter{Hfootnote}
4228   \RequirePackage{omdoc}
4229 }

```

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

```

4230 \RequirePackage{mikoslides}
4231 </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.

```

4232 \*package>
4233 \bool_if:NT \c__mikoslides_notes_bool {
4234   \RequirePackage{a4wide}
4235   \RequirePackage{marginnote}
4236   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4237   \RequirePackage{mdframed}
4238   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4239   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4240 }
4241 \RequirePackage{stex-compatibility}
4242 \RequirePackage{stex-tikzinput}
4243 \RequirePackage{etoolbox}
4244 \RequirePackage{amssymb}
4245 \RequirePackage{amsmath}
4246 \RequirePackage{comment}
4247 \RequirePackage{textcomp}
4248 \RequirePackage{url}
4249 \RequirePackage{graphicx}
4250 \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

```

4251 \bool_if:NT \c__mikoslides_notes_bool {
4252   \renewcommand\usetheme[2][]{\usepackage[#1]{beamernotestheme#2}}
4253 }

```

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

```

4254 \newcounter{slide}
4255 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4256 \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.

```

4257 \bool_if:NTF \c__mikoslides_notes_bool {
4258   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4259 }{
4260   \excludecomment{note}
4261 }

```

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.

```
4262 \bool_if:NT \c__mikoslides_notes_bool {
4263   \newlength{\slideframewidth}
4264   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4265 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4266   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4267     \bool_set_true:N #1
4268   }{
4269     \bool_set_false:N #1
4270   }
4271 }
4272 \keys_define:nn{mikoslides / frame}{
4273   label .str_set_x:N = \l__mikoslides_frame_label_str,
4274   allowframebreaks .code:n = {
4275     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4276   },
4277   allowdisplaybreaks .code:n = {
4278     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4279   },
4280   fragile .code:n = {
4281     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4282   },
4283   shrink .code:n = {
4284     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4285   },
4286   squeeze .code:n = {
4287     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4288   },
4289   t .code:n = {
4290     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4291   },
4292 }
4293 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4294   \str_clear:N \l__mikoslides_frame_label_str
4295   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4296   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4297   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4298   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4299   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4300   \bool_set_true:N \l__mikoslides_frame_t_bool
4301   \keys_set:nn { mikoslides / frame }{ #1 }
4302 }
```

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

```
4303 \renewenvironment{frame}[1][]{
4304   \__mikoslides_frame_args:n{#1}
4305   \sffamily
4306   \stepcounter{slide}
4307   \def\@currentlabel{\theslide}
4308   \str_if_empty:NF \l__mikoslides_frame_label_str {
4309     \label{\l__mikoslides_frame_label_str}
```

```
4310 }
```

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

```
4311 \def\itemize@level{outer}
4312 \def\itemize@outer{outer}
4313 \def\itemize@inner{inner}
4314 \renewcommand\newpage{\addtocounter{framenum}{1}}
4315 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4316 \renewenvironment{itemize}{
4317   \ifx\itemize@level\itemize@outer
4318     \def\itemize@label{\$ \rhd \$}
4319   \fi
4320   \ifx\itemize@level\itemize@inner
4321     \def\itemize@label{\$ \scriptstyle \rhd \$}
4322   \fi
4323   \begin{list}
4324     {\itemize@label}
4325     {\setlength{\labelsep}{.3em}
4326      \setlength{\labelwidth}{.5em}
4327      \setlength{\leftmargin}{1.5em}
4328     }
4329   \edef\itemize@level{\itemize@inner}
4330 }{
4331   \end{list}
4332 }
```

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

```
4333 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4334 }{
4335   \medskip\miko@slidelabel\end{mdframed}
4336 }
```

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

`\frametitle`

```
4337 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4338 }
```

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

EdN:18

`\pause` 18

```
4339 \bool_if:NT \c__mikoslides_notes_bool {
4340   \newcommand\pause{}
4341 }
```

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

`nomtext`

```
4342 \bool_if:NTF \c__mikoslides_notes_bool {
4343   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}
4344 }{
4345   \excludecomment{nomtext}
4346 }
```

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

nomgroup

```
4347 \bool_if:NTF \c__mikoslides_notes_bool {  
4348   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
4349 }{  
4350   \excludecomment{nomgroup}  
4351 }
```

ndefinition

```
4352 \bool_if:NTF \c__mikoslides_notes_bool {  
4353   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}  
4354 }{  
4355   \excludecomment{ndefinition}  
4356 }
```

nassertion

```
4357 \bool_if:NTF \c__mikoslides_notes_bool {  
4358   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}  
4359 }{  
4360   \excludecomment{nassertion}  
4361 }
```

nsproof

```
4362 \bool_if:NTF \c__mikoslides_notes_bool {  
4363   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
4364 }{  
4365   \excludecomment{nsproof}  
4366 }
```

nexample

```
4367 \bool_if:NTF \c__mikoslides_notes_bool {  
4368   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}  
4369 }{  
4370   \excludecomment{nexample}  
4371 }
```

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

```
4372 \def\inputref@preskip{\smallskip}  
4373 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
4374 \let\orig@inputref\inputref  
4375 \def\inputref{\@ifstar\ninputref\orig@inputref}  
4376 \newcommand\ninputref[2] [] {  
4377   \bool_if:NT \c__mikoslides_notes_bool {  
4378     \orig@inputref[#1]{#2}  
4379   }  
4380 }
```

(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>}`.

```

4381 \newlength{\slidelogoheight}
4382
4383 \bool_if:NTF \c__mikoslides_notes_bool {
4384   \setlength{\slidelogoheight}{.4cm}
4385 }{
4386   \setlength{\slidelogoheight}{1cm}
4387 }
4388 \newsavebox{\slidelogo}
4389 \sbox{\slidelogo}{\TeX}
4390 \newrobustcmd{\setslidelogo}[1]{
4391   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4392 }
```

(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.

```

4393 \def\source{Michael Kohlhase}% customize locally
4394 \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.

```

4395 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4396 \newsavebox{\cclogo}
4397 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4398 \newif\ifcchref\cchreffalse
4399 \AtBeginDocument{
4400   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4401 }
4402 \def\licensing{
4403   \ifcchref
4404     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4405   \else
4406     {\usebox{\cclogo}}
4407   \fi
4408 }
4409 \newrobustcmd{\setlicensing}[2][]{
4410   \def\@url{#1}
4411   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4412   \ifx\@url\@empty
4413     \def\licensing{{\usebox{\cclogo}}}
4414   \else
4415     \def\licensing{
```



```

4416     \ifcchref
4417     \href{#1}{\usebox{\cclogo}}
4418   \else
4419     {\usebox{\cclogo}}
4420   \fi
4421 }
4422 \fi
4423 }

```

(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

```

4424 \newrobustcmd\miko@slidelabel{
4425   \vbox to \slidelogoheight{
4426     \vss\hbox to \slidewidth
4427     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4428   }
4429 }

```

(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.

```

4430 \def\Gin@mhrepos{}
4431 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4432 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4433 \newrobustcmd\frameimage[2][]{
4434   \stepcounter{slide}
4435   \bool_if:NT \c__mikoslides_frameimages_bool {
4436     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4437     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4438     \begin{center}
4439       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4440         \fbox{
4441           \ifx\Gin@ewidth\@empty
4442             \ifx\Gin@mhrepos\@empty
4443               \mhgraphics[width=\slidewidth,#1]{#2}
4444             \else
4445               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4446             \fi
4447           \else% Gin@ewidth empty
4448             \ifx\Gin@mhrepos\@empty
4449               \mhgraphics[#1]{#2}
4450             \else
4451               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4452             \fi
4453           \fi% Gin@ewidth empty
4454         }
4455       }{
4456         \ifx\Gin@ewidth\@empty

```

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

```

4457         \ifx\Gin@mhrepos\empty
4458             \mhgraphics[width=\slidewidth,#1]{#2}
4459         \else
4460             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4461         \fi
4462         \ifx\Gin@mhrepos\empty
4463             \mhgraphics[#1]{#2}
4464         \else
4465             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4466         \fi
4467     \fi% Gin@ewidth empty
4468 }
4469 \end{center}
4470 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4471 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4472 }
4473 } % 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.

```

4474 \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

```

4475 \AddToHook{begindocument}{
4476     \definecolor{green}{rgb}{0,.5,0}
4477     \definecolor{purple}{cmyk}{.3,1,0,.17}
4478 }

```

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`.

```

4479 % \def\STpresent#1{\textcolor{blue}{#1}}
4480 \def\defemph#1{\textcolor{magenta}{#1}}
4481 \def\symrefemph#1{\textcolor{cyan}{#1}}
4482 \def\compemph#1{\textcolor{blue}{#1}}
4483 \def\titleemph#1{\textcolor{blue}{#1}}
4484 \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.

```

4485 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4486 \def\smalltextwarning{
4487     \pgfuseimage{miko@small@dbend}
4488     \xspace
4489 }
4490 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4491 \newrobustcmd\textwarning{
4492   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4493   \xspace
4494 }
4495 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4496 \newrobustcmd\bigtextwarning{
4497   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4498   \xspace
4499 }

```

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

```

4500 \newrobustcmd\putgraphicsat[3]{
4501   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4502 }
4503 \newrobustcmd\putat[2]{
4504   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4505 }

```

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`.

```

4506 \bool_if:NT \c__mikoslides_sectocframes_bool {
4507   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4508     \newcounter{chapter}\counterwithin*{section}{chapter}
4509   }{
4510     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4511       \newcounter{chapter}\counterwithin*{section}{chapter}
4512     }
4513   }
4514 }

```

`\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
4515 \def\part@prefix{}
4516 \@ifpackageloaded{omdoc}{}{
4517   \str_case:VnF \__mikoslidestopsect {
4518     {part}{
4519       \int_set:Nn \l_document_structure_section_level_int {0}
4520       \def\thesection{\arabic{chapter}.\arabic{section}}
4521       \def\part@prefix{\arabic{chapter}.}
4522     }
4523     {chapter}{
4524       \int_set:Nn \l_document_structure_section_level_int {1}
4525       \def\thesection{\arabic{chapter}.\arabic{section}}
4526       \def\part@prefix{\arabic{chapter}.}
4527     }
4528   }{
4529     \int_set:Nn \l_document_structure_section_level_int {2}
4530     \def\part@prefix{}

```

```

4531 }
4532 }
4533
4534 \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

```

4535 \renewenvironment{omgroup}[2][]{
4536   \__document_structure_omgroup_args:n { #1 }
4537   \int_incr:N \l_document_structure_omgroup_level_int
4538   \int_incr:N \l_document_structure_section_level_int
4539   \bool_if:NT \c__mikoslides_sectocframes_bool {
4540     \stepcounter{slide}
4541     \begin{frame}[noframenumbering]
4542     \vfill\Large\centering
4543     \red{
4544       \ifcase\l_document_structure_section_level_int\or
4545         \stepcounter{part}
4546         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4547         \def\currentsectionlevel{\omdoc@part@kw}
4548       \or
4549         \stepcounter{chapter}
4550         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4551         \def\currentsectionlevel{\omdoc@chapter@kw}
4552       \or
4553         \stepcounter{section}
4554         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4555         \def\currentsectionlevel{\omdoc@section@kw}
4556       \or
4557         \stepcounter{subsection}
4558         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4559         \def\currentsectionlevel{\omdoc@subsection@kw}
4560       \or
4561         \stepcounter{subsubsection}
4562         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4563         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4564       \or
4565         \stepcounter{mparagraph}
4566         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4567         \def\currentsectionlevel{\omdoc@paragraph@kw}
4568       \fi% end ifcase
4569       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4570       \quad #2%
4571     }%
4572     \vfill%
4573     \end{frame}%
4574   }
4575   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4576 }{}
4577 }

```

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

```

4578 \def\inserttheorembodyfont{\normalfont}
4579 \bool_if:NF \c__mikoslides_notes_bool {
4580   \defbeamertemplate{theorem begin}{miko}
4581   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4582     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4583     \inserttheorempunctuation\inserttheorembodyfont\space}
4584   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4585   \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.

```

4586   \expandafter\def\csname Parent2\endcsname{}
4587 }
4588 \bool_if:NT \c__mikoslides_notes_bool {
4589   \renewenvironment{columns}[1][]{%
4590     \par\noindent%
4591     \begin{minipage}%
4592       \slidewidth\centering\leavevmode%
4593   }{%
4594     \end{minipage}\par\noindent%
4595   }%
4596   \newsavebox\columnbox%
4597   \renewenvironment<>{column}[2][]{%
4598     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4599   }{%
4600     \end{minipage}\end{lrbox}\usebox\columnbox%
4601   }%
4602 }
4603 \bool_if:NTF \c__mikoslides_noproblems_bool {
4604   \newenvironment{problems}{}{}
4605 }{
4606   \excludecomment{problems}
4607 }
```

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.

```

4608 \gdef\printexcursions{}
4609 \newcommand\excursionref[2]{% label, text
4610   \bool_if:NT \c__mikoslides_notes_bool {
4611     \begin{omtext}[title=Excursion]
4612       #2 \sref[fallback=the appendix]{#1}.
4613     \end{omtext}
4614   }
4615 }
4616 \newcommand\activate@excursion[2][{}{
4617   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4618 }
4619 \newcommand\excursion[4][{}]{% repos, label, path, text
4620   \bool_if:NT \c__mikoslides_notes_bool {
4621     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4622   }
4623 }

```

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

\excursiongroup

```

4624 \keys_define:nn{mikoslides / excursiongroup }{
4625   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4626   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4627   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4628 }
4629 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4630   \tl_clear:N \l__mikoslides_excursion_intro_tl
4631   \str_clear:N \l__mikoslides_excursion_id_str
4632   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4633   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4634 }
4635 \newcommand\excursiongroup[1][{}]{
4636   \__mikoslides_excursion_args:n{ #1 }
4637   \ifdefempty\printexcursions{}% only if there are excursions
4638   {\begin{note}
4639     \begin{omgroup}[#1]{Excursions}%
4640       \ifdefempty\l__mikoslides_excursion_intro_tl{{
4641         \inputref[\l__mikoslides_excursion_mhrepos_str]{
4642           \l__mikoslides_excursion_intro_tl
4643         }
4644       }
4645       \printexcursions%
4646       \end{omgroup}
4647     \end{note}}
4648   }
4649 }

```

(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.

```
4650 <*package>
4651 <@@=problems>
4652 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4653 \RequirePackage{l3keys2e,expl-keystr-compatible}
4654
4655 \keys_define:nn { problem / pkg }{
4656   notes      .default:n    = { true },
4657   notes      .bool_set:N   = \c__problems_notes_bool,
4658   gnotes     .default:n    = { true },
4659   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4660   hints      .default:n    = { true },
4661   hints      .bool_set:N   = \c__problems_hints_bool,
4662   solutions  .default:n    = { true },
4663   solutions  .bool_set:N   = \c__problems_solutions_bool,
4664   pts        .default:n    = { true },
4665   pts        .bool_set:N   = \c__problems_pts_bool,
4666   min        .default:n    = { true },
4667   min        .bool_set:N   = \c__problems_min_bool,
4668   boxed      .default:n    = { true },
4669   boxed      .bool_set:N   = \c__problems_boxed_bool,
4670   unknown    .code:n       = {}
4671 }
4672 \def\solutionstrue{
4673   \bool_set_true:N \c__problems_solutions_bool
4674 }
4675 \def\solutionsfalse{
4676   \bool_set_false:N \c__problems_solutions_bool
4677 }
4678
4679 \ProcessKeysOptions{ problem / pkg }
```

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

```

4680 \RequirePackage{stex-compatibility}
4681 \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.

```

4682 \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.

```

4683 \def\prob@problem@kw{Problem}
4684 \def\prob@solution@kw{Solution}
4685 \def\prob@hint@kw{Hint}
4686 \def\prob@note@kw{Note}
4687 \def\prob@gnote@kw{Grading}
4688 \def\prob@pt@kw{pt}
4689 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4690 \@ifpackageloaded{babel}{
4691   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4692   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4693     \input{problem-ngerman.ldf}
4694   }
4695   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4696     \input{problem-finnish.ldf}
4697   }
4698   \clist_if_in:NnT \l_tmpa_clist {french}{
4699     \input{problem-french.ldf}
4700   }
4701   \clist_if_in:NnT \l_tmpa_clist {russian}{
4702     \input{problem-russian.ldf}
4703   }
4704 }{}

```

33.2 Problems and Solutions

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

```

4705 \keys_define:nn{ problem / problem }{
4706   id      .str_set:x:N = \l__problems_prob_id_str,
4707   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4708   min     .tl_set:N    = \l__problems_prob_min_tl,
4709   title   .tl_set:N    = \l__problems_prob_title_tl,
4710   refnum  .int_set:N   = \l__problems_prob_refnum_int
4711 }
4712 \cs_new_protected:Nn \__problems_prob_args:n {
4713   \str_clear:N \l__problems_prob_id_str
4714   \tl_clear:N \l__problems_prob_pts_tl
4715   \tl_clear:N \l__problems_prob_min_tl
4716   \tl_clear:N \l__problems_prob_title_tl

```



```

4717 \int_zero_new:N \l__problems_prob_refnum_int
4718 \keys_set:nn { problem / problem }{ #1 }
4719 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4720   \let\l__problems_inclprob_refnum_int\undefined
4721 }
4722 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4723 \newcounter{problem}
4724 \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.

```

4725 \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

```

4726 \newcommand\prob@number{
4727   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4728     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4729   }{
4730     \int_if_exist:NTF \l__problems_prob_refnum_int {
4731       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4732     }{
4733       \prob@label\theproblem
4734     }
4735   }
4736 }

```

(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.

```

4737 \newcommand\prob@title[3]{%
4738   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4739     #2 \l__problems_inclprob_title_tl #3
4740   }{
4741     \tl_if_exist:NTF \l__problems_prob_title_tl {
4742       #2 \l__problems_prob_title_tl #3
4743     }{
4744       #1
4745     }
4746   }
4747 }

```

(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.

```

4748 \def\prob@heading{
4749   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4750   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4751 }

```

(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`

```

4752 \newenvironment{problem}[1][1]{
4753   \_problems_prob_args:n{#1}%\sref@target%
4754   \@in@omtexttrue% we are in a statement (for inline definitions)
4755   \stepcounter{problem}\record@problem
4756   \def\current@section@level{\prob@problem@kw}
4757   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4758 }%
4759   {\smallskip}
4760   \bool_if:NT \c__problems_boxed_bool {
4761     \surroundwithmdframed{problem}
4762   }

```

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

```

4763 \def\record@problem{
4764   \protected@write\@auxout{}
4765   {
4766     \string\@problem{\prob@number}
4767     {
4768       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4769         \l__problems_inclprob_pts_tl
4770       }{
4771         \l__problems_prob_pts_tl
4772       }
4773     }%
4774     {
4775       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4776         \l__problems_inclprob_min_tl
4777       }{
4778         \l__problems_prob_min_tl
4779       }
4780     }
4781   }
4782 }

```

(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).

```

4783 \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.

```

4784 \keys_define:nn { problem / solution }{
4785   id          .str_set_x:N = \l__problems_solution_id_str ,
4786   for         .tl_set:N    = \l__problems_solution_for_tl ,
4787   height      .dim_set:N   = \l__problems_solution_height_dim ,
4788   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4789   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4790   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4791 }
4792 \cs_new_protected:Nn \__problems_solution_args:n {
4793   \str_clear:N \l__problems_solution_id_str
4794   \tl_clear:N \l__problems_solution_for_tl
4795   \tl_clear:N \l__problems_solution_srccite_tl
4796   \clist_clear:N \l__problems_solution_creators_clist
4797   \clist_clear:N \l__problems_solution_contributors_clist
4798   \dim_zero:N \l__problems_solution_height_dim
4799   \keys_set:nn { problem / solution }{ #1 }
4800 }

```

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

```

4801 \newcommand\@startsolution[1][ ]{
4802   \__problems_solution_args:n { #1 }
4803   \@in@omtexttrue% we are in a statement.
4804   \bool_if:NF \c__problems_boxed_bool { \hrule }
4805   \smallskip\noindent
4806   {\textbf\prob@solution@kw : \enspace}
4807   \begin{small}
4808   \def\current@section@level{\prob@solution@kw}
4809   \ignorespacesandpars
4810 }

```

\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.

```

4811 \newcommand\startsolutions{
4812   \specialcomment{solution}{\@startsolution}{
4813     \bool_if:NF \c__problems_boxed_bool {
4814       \hrule\medskip
4815     }
4816     \end{small}%
4817   }
4818   \bool_if:NT \c__problems_boxed_bool {
4819     \surroundwithmdframed{solution}
4820   }
4821 }

```

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

\stopsolutions

```

4822 \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.

```

4823 \bool_if:NTF \c_problems_solutions_bool {
4824   \startsolutions
4825 }{
4826   \stopsolutions
4827 }

```

exnote

```

4828 \bool_if:NTF \c_problems_notes_bool {
4829   \newenvironment{exnote}[1][]{
4830     \par\smallskip\hrule\smallskip
4831     \noindent\textbf{\prob@note@kw : }\small
4832   }{
4833     \smallskip\hrule
4834   }
4835 }{
4836   \excludecomment{exnote}
4837 }

```

hint

```

4838 \bool_if:NTF \c_problems_notes_bool {
4839   \newenvironment{hint}[1][]{
4840     \par\smallskip\hrule\smallskip
4841     \noindent\textbf{\prob@hint@kw :~ }\small
4842   }{
4843     \smallskip\hrule
4844   }
4845   \newenvironment{exhint}[1][]{
4846     \par\smallskip\hrule\smallskip
4847     \noindent\textbf{\prob@hint@kw :~ }\small
4848   }{
4849     \smallskip\hrule
4850   }
4851 }{
4852   \excludecomment{hint}
4853   \excludecomment{exhint}
4854 }

```

gnote

```

4855 \bool_if:NTF \c_problems_notes_bool {
4856   \newenvironment{gnote}[1][]{
4857     \par\smallskip\hrule\smallskip
4858     \noindent\textbf{\prob@gnote@kw : }\small
4859   }{
4860     \smallskip\hrule
4861   }
4862 }{
4863   \excludecomment{gnote}
4864 }

```

33.3 Multiple Choice Blocks

```

4865 \newenvironment{mcb}{
4866   \begin{enumerate}
4867 }{
4868   \end{enumerate}
4869 }

```

we define the keys for the mcc macro

```

4870 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4871   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4872     \bool_set_true:N #1
4873   }{
4874     \bool_set_false:N #1
4875   }
4876 }
4877 \keys_define:nn { problem / mcc }{
4878   id          .str_set:x:N = \l__problems_mcc_id_str ,
4879   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4880   T           .default:n   = { true } ,
4881   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4882   F           .default:n   = { true } ,
4883   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4884   Ttext       .code:n      = {
4885     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4886   } ,
4887   Ftext       .code:n      = {
4888     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4889   }
4890 }
4891 \cs_new_protected:Nn \l__problems_mcc_args:n {
4892   \str_clear:N \l__problems_mcc_id_str
4893   \tl_clear:N \l__problems_mcc_feedback_tl
4894   \bool_set_true:N \l__problems_mcc_t_bool
4895   \bool_set_true:N \l__problems_mcc_f_bool
4896   \bool_set_true:N \l__problems_mcc_Ttext_bool
4897   \bool_set_false:N \l__problems_mcc_Ftext_bool
4898   \keys_set:nn { problem / mcc }{ #1 }
4899 }

```

\mcc

```

4900 \newcommand\mcc[2][] {
4901   \l__problems_mcc_args:n{ #1 }
4902   \item #2
4903   \bool_if:NT \c__problems_solutions_bool {
4904     \
4905     \bool_if:NT \l__problems_mcc_t_bool {
4906       % TODO!
4907       % \ifcsstring{mcc@T}{T}{ }\{ \mcc@Ttext }%
4908     }
4909     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

4910      % TODO!
4911      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4912    }
4913    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4914      !
4915    }{
4916      \l__problems_mcc_feedback_tl
4917    }
4918  }
4919 } %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.

```

4920
4921 \keys_define:nn{ problem / inclproblem }{
4922   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4923   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4924   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4925   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4926   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4927   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4928 }
4929 \cs_new_protected:Nn \l__problems_inclprob_args:n {
4930   % \str_clear:N \l__problems_prob_id_str
4931   \tl_clear:N \l__problems_inclprob_pts_tl
4932   \tl_clear:N \l__problems_inclprob_min_tl
4933   \tl_clear:N \l__problems_inclprob_title_tl
4934   \int_zero_new:N \l__problems_inclprob_refnum_int
4935   \str_clear:N \l__problems_inclprob_mhrepos_str
4936   \keys_set:nn { problem / inclproblem }{ #1 }
4937   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4938     \let\l__problems_inclprob_pts_tl\undefined
4939   }
4940   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4941     \let\l__problems_inclprob_min_tl\undefined
4942   }
4943   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4944     \let\l__problems_inclprob_title_tl\undefined
4945   }
4946   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4947     \let\l__problems_inclprob_refnum_int\undefined
4948   }
4949 }
4950
4951 \cs_new_protected:Nn \l__problems_inclprob_clear: {
4952   % \str_clear:N \l__problems_prob_id_str
4953   \let\l__problems_inclprob_pts_tl\undefined
4954   \let\l__problems_inclprob_min_tl\undefined

```

```

4955 \let\l__problems_inclprob_title_tl\undefined
4956 \let\l__problems_inclprob_refnum_int\undefined
4957 \let\l__problems_inclprob_mhrepos_str\undefined
4958 }
4959
4960 \newcommand\includeproblem[2][ ]{
4961   \__problems_inclprob_args:n{ #1 }
4962   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4963     \input{#2}
4964   }{
4965     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4966       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4967     }
4968   }
4969   \__problems_inclprob_clear:
4970 }

```

(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.

```

4971 \AddToHook{enddocument}{
4972   \bool_if:NT \c__problems_pts_bool {
4973     \message{Total:~\arabic{pts}~points}
4974   }
4975   \bool_if:NT \c__problems_min_bool {
4976     \message{Total:~\arabic{min}~minutes}
4977   }
4978 }

```

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

```

4979 \def\pts#1{
4980   \bool_if:NT \c__problems_pts_bool {
4981     \marginpar{#1~\prob@pt@kw}
4982   }
4983 }
4984 \def\min#1{
4985   \bool_if:NT \c__problems_min_bool {
4986     \marginpar{#1~\prob@min@kw}
4987   }
4988 }

```

\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.

```

4989 \newcounter{pts}
4990 \def\show@pts{
4991   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4992     \bool_if:NT \c__problems_pts_bool {
4993       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4994       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

4995     }
4996   }{
4997     \tl_if_exist:NT \l__problems_prob_pts_tl {
4998       \bool_if:NT \c__problems_pts_bool {
4999         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5000         \addtocounter{pts}{\l__problems_prob_pts_tl}
5001       }
5002     }
5003   }
5004 }

```

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

\show@min

```

5005 \newcounter{min}
5006 \def\show@min{
5007   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5008     \bool_if:NT \c__problems_min_bool {
5009       \marginpar{\l__problems_inclprob_pts_tl;min}
5010       \addtocounter{min}{\l__problems_inclprob_min_tl}
5011     }
5012   }{
5013     \tl_if_exist:NT \l__problems_prob_min_tl {
5014       \bool_if:NT \c__problems_min_bool {
5015         \marginpar{\l__problems_prob_min_tl;min}
5016         \addtocounter{min}{\l__problems_prob_min_tl}
5017       }
5018     }
5019   }
5020 }
5021 \</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.

```
5022 <@@=hwexam>
5023 <*cls>
5024 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5025 \RequirePackage{l3keys2e,expl-keystr-compatible}
5026 \DeclareOption*{
5027   \PassOptionsToClass{\CurrentOption}{omdoc}
5028   \PassOptionsToPackage{\CurrentOption}{stex}
5029   \PassOptionsToPackage{\CurrentOption}{hwexam}
5030   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5031 }
5032 \ProcessOptions
```

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

```
5033 \LoadClass{omdoc}
5034 \RequirePackage{stex}
5035 \RequirePackage{hwexam}
5036 \RequirePackage{tikzinput}
5037 \RequirePackage{graphicx}
5038 \RequirePackage{a4wide}
5039 \RequirePackage{amssymb}
5040 \RequirePackage{amstext}
5041 \RequirePackage{amsmath}
```

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

```

5042 \newcommand\assig@default@type{\hwexam@assignment@kw}
5043 \def\document@hwexamtype{\assig@default@type}
5044 <@@=document_structure>
5045 \keys_define:nn { document-structure / document }{
5046 id .str_set_x:N = \c_document_structure_document_id_str,
5047 hwexamtype .tl_set:N = \document@hwexamtype
5048 }
5049 <@@=hwexam>
5050 </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.

```
5051 \*package>
5052 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5053 \RequirePackage{l3keys2e,expl-keystr-compat}
5054
5055 \newif\iftest\testfalse
5056 \DeclareOption{test}{\testtrue}
5057 \newif\ifmultiple\multiplefalse
5058 \DeclareOption{multiple}{\multipletrue}
5059 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5060 \ProcessOptions
```

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

```
5061 \RequirePackage{keyval}[1997/11/10]
5062 \RequirePackage{problem}
```

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

```
5063 \newcommand\hwexam@assignment@kw{Assignment}
5064 \newcommand\hwexam@given@kw{Given}
5065 \newcommand\hwexam@due@kw{Due}
5066 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5067   space}%
5068 \newcommand\correction@probs@kw{prob.}%
5069 \newcommand\correction@pts@kw{total}%
5070 \newcommand\correction@reached@kw{reached}%
5071 \newcommand\correction@sum@kw{Sum}%
5072 \newcommand\correction@grade@kw{grade}%
5073 \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

```

5074 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5075
5076 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5077 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5078   \input{hwexam-ngerman.ldf}
5079 }
5080 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5081   \input{hwexam-finnish.ldf}
5082 }
5083 \clist_if_in:NnT \l_tmpa_clist {french}{
5084   \input{hwexam-french.ldf}
5085 }
5086 \clist_if_in:NnT \l_tmpa_clist {russian}{
5087   \input{hwexam-russian.ldf}
5088 }

```

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.

```

5089 \newcounter{assignment}
5090 \numberproblemsin{assignment}
5091 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5092 \keys_define:nn { hwexam / assignment } {
5093   id .str_set:N = \l__hwexam_assign_id_str,
5094   number .int_set:N = \l__hwexam_assign_number_int,
5095   title .tl_set:N = \l__hwexam_assign_title_tl,
5096   type .tl_set:N = \l__hwexam_assign_type_tl,
5097   given .tl_set:N = \l__hwexam_assign_given_tl,
5098   due .tl_set:N = \l__hwexam_assign_due_tl,
5099   loadmodules .code:n = {
5100     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5101   }
5102 }
5103 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5104   \str_clear:N \l__hwexam_assign_id_str
5105   \int_set:Nn \l__hwexam_assign_number_int {-1}
5106   \tl_clear:N \l__hwexam_assign_title_tl
5107   \tl_clear:N \l__hwexam_assign_type_tl
5108   \tl_clear:N \l__hwexam_assign_given_tl
5109   \tl_clear:N \l__hwexam_assign_due_tl
5110   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5111   \keys_set:nn { hwexam / assignment }{ #1 }
5112 }

```

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.

```

5113 \newcommand\given@due[2]{
5114 \bool_lazy_all:nF {
5115 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5116 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5117 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5118 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5119 }{ #1 }
5120
5121 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5122 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5123 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5124 }
5125 }{
5126 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5127 }
5128
5129 \bool_lazy_or:nnF {
5130 \bool_lazy_and_p:nn {
5131 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5132 }{
5133 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5134 }
5135 }{
5136 \bool_lazy_and_p:nn {
5137 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5138 }{
5139 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5140 }
5141 }{ ,~ }
5142
5143 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5144 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5145 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5146 }
5147 }{
5148 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5149 }
5150
5151 \bool_lazy_all:nF {
5152 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5153 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5154 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5155 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5156 }{ #2 }
5157 }

```

`\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.

```

5158 \newcommand\assignment@title[3]{

```

```

5159 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5160 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5161 #1
5162 }{
5163 #2\l__hwexam_assign_title_tl#3
5164 }
5165 }{
5166 #2\l__hwexam_inclassassign_title_tl#3
5167 }
5168 }

```

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

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

```

5169 \newcommand\assignment@number{
5170 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5171 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5172 \int_use:N \l__hwexam_assign_number_int
5173 }
5174 }{
5175 \int_use:N \l__hwexam_inclassassign_number_int
5176 }
5177 }

```

(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.

```

5178 \newenvironment{assignment}[1][]{
5179 \__hwexam_assignment_args:n { #1 }
5180 %\sref@target
5181 \let\__hwexamnum\l__hwexam_assign_number_int
5182 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5183 \stepcounter{assignment}
5184 }{
5185 \setcounter{assignment}{\int_use:N\__hwexamnum}
5186 }
5187 \setcounter{problem}{0}
5188 \def\current@section@level{\document@hwexamtype}
5189 %\sref@label@id{\document@hwexamtype \thesection}
5190 \begin{@assignment}
5191 }{
5192 \end{@assignment}
5193 }

```

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

```

5194 \def\__hwexasstitle{
5195 \protect\document@hwexamtype~\arabic{assignment}
5196 \assignment@title{}\;{} \; -- \given@due{}\}
5197 }

```

```

5198 \ifmultiple
5199 \newenvironment{@assignment}{
5200 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5201 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5202 }{
5203 \begin{omgroup}{\__hwexasstitle}
5204 }
5205 }{
5206 \end{omgroup}
5207 }

```

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

```

5208 \else
5209 \newenvironment{@assignment}{
5210 \begin{center}\bf
5211 \Large\@title\strut\
5212 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5213 \large\given@due{--\;}{\;}{--}
5214 \end{center}
5215 }{}
5216 \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.

```

5217 \keys_define:nn { hwexam / inclassignment } {
5218 %id .str_set_x:N = \l__hwexam_assign_id_str,
5219 number .int_set:N = \l__hwexam_inclassign_number_int,
5220 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5221 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5222 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5223 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5224 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5225 }
5226 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5227 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5228 \tl_clear:N \l__hwexam_inclassign_title_tl
5229 \tl_clear:N \l__hwexam_inclassign_type_tl
5230 \tl_clear:N \l__hwexam_inclassign_given_tl
5231 \tl_clear:N \l__hwexam_inclassign_due_tl
5232 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5233 \keys_set:nn { hwexam / inclassignment }{ #1 }
5234 }
5235 \__hwexam_inclassignment_args:n {}
5236
5237 \newcommand\inputassignment[2][ ]{
5238 \__hwexam_inclassignment_args:n { #1 }
5239 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5240 \input{#2}
5241 }{
5242 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5243 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5244 }
5245 }
5246 \__hwexam_inclasssign_args:n {}
5247 }
5248 \newcommand\includeassignment[2][ ]{
5249 \newpage
5250 \inputassignment[#1]{#2}
5251 }

```

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

35.4 Typesetting Exams

\quizheading

```

5252 \ExplSyntaxOff
5253 \newcommand\quizheading[1]{%
5254 \def\@tas{#1}%
5255 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5256 \ifx\@tas\empty\else%
5257 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5258 \fi%
5259 }
5260 \ExplSyntaxOn

```

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

\testheading

```

5261 \keys_define:nn { hwexam / testheading } {
5262 min .tl_set:N = \l__hwexam_testheading_min_tl,
5263 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5264 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5265 }
5266 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5267 \tl_clear:N \l__hwexam_testheading_min_tl
5268 \tl_clear:N \l__hwexam_testheading_duration_tl
5269 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5270 \keys_set:nn { hwexam / testheading }{ #1 }
5271 }
5272 \newenvironment{testheading}[1][ ]{
5273 \__hwexam_testheading_args:n{ #1 }
5274 \noindent\large{Name:~\hfill
5275 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5276 \begin{center}
5277 \Large\textbf{\@title}\[1ex]
5278 \large\@date\[3ex]
5279 \end{center}
5280 \textbf{You~have~
5281 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5282 \l__hwexam_testheading_min_tl~minutes
5283 }{
5284 \l__hwexam_testheading_duration_tl
5285 }~

```



```

5286 (sharp)~for~the~test
5287 };\
5288 Write~the~solutions~to~the~sheet.
5289 \par\noindent
5290 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5291 \advance\check@time by -\theassignment@totalmin
5292 The~estimated~time~for~solving~this~exam~is~
5293 {\theassignment@totalmin}~minutes,~
5294 leaving~you~{\the\check@time}~minutes~for~revising~
5295 your~exam.
5296
5297 \par\noindent
5298 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5299 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5300 You~can~reach~{\theassignment@totalpts}~points~if~you~
5301 solve~all~problems.~You~will~only~need~
5302 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5303 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
5304 \vfill
5305 \begin{center}
5306 {
5307 \Large\em You~have~ample~time,~so~take~it~slow~
5308 and~avoid~rushing~to~mistakes!\}[2ex]
5309 Different~problems~test~different~skills~and~
5310 knowledge,~so~do~not~get~stuck~on~one~problem.
5311 }
5312 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5313 \end{center}
5314 }{
5315 \newpage
5316 }

```

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

\testspace

```

5317 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5318 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5319 \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.

```

5320 <@=problems>
5321 \renewcommand\@problem[3]{
5322 \stepcounter{assignment@probs}
5323 \def\__problemspts{#2}

```

```

5324 \ifx\__problemspts\@empty\else
5325 \addtocounter{assignment@totalpts}{#2}
5326 \fi
5327 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5328 \xdef\correction@probs{\correction@probs & #1}%
5329 \xdef\correction@pts{\correction@pts & #2}
5330 \xdef\correction@reached{\correction@reached &}
5331 }
5332 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5333 \newcounter{assignment@probs}
5334 \newcounter{assignment@totalpts}
5335 \newcounter{assignment@totalmin}
5336 \def\correction@probs{\correction@probs@kw}%
5337 \def\correction@pts{\correction@pts@kw}%
5338 \def\correction@reached{\correction@reached@kw}%
5339 \def\after@correction@table{}%
5340 \stepcounter{assignment@probs}
5341 \newcommand\correction@table{
5342 \resizebox{\textwidth}{!}{%
5343 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5344 &\multicolumn{\theassignment@probs}{c|}|%|
5345 {\footnotesize\correction@forgrading@kw} &\\ \hline
5346 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5347 \correction@pts & \theassignment@totalpts & \\ \hline
5348 \correction@reached & & \[.7cm]\hline
5349 \end{tabular}}
5350 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5351 \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/wierfont
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

\newcommand\wierfont{\font\wierfont\char65}
\newcommand\denkerfont{\font\denkerfont\char65}
\newcommand\uhrfont{\font\uhrfont\char65}
\newcommand\warnschildfont{\font\warnschildfont\char 65}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\wierfont}

```