

The \TeX 3 Package *

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

2022-01-27

Abstract

TODO

*Version 3.0 (last revised 2022-01-27)

Contents

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

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

16	problem.sty: An Infrastructure for formatting Problems	51
16.1	Introduction	51
16.2	The User Interface	51
16.2.1	Package Options	51
16.2.2	Problems and Solutions	52
16.2.3	Multiple Choice Blocks	53
16.2.4	Including Problems	53
16.2.5	Reporting Metadata	53
16.3	Limitations	53
17	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	55
17.1	Introduction	56
17.2	The User Interface	56
17.2.1	Package and Class Options	56
17.2.2	Assignments	56
17.2.3	Typesetting Exams	56
17.2.4	Including Assignments	57
17.3	Limitations	57
IV	Implementation	59
18	gTeX-Basics Implementation	60
18.1	The gTeXDocument Class	60
18.2	Preliminaries	60
18.3	Messages and logging	61
18.4	Persistence	62
18.5	HTML Annotations	62
18.6	Languages	65
18.7	Activating/Deactivating Macros	66
19	gTeX-MathHub Implementation	68
19.1	Generic Path Handling	68
19.2	PWD and kpsewhich	70
19.3	File Hooks and Tracking	71
19.4	MathHub Repositories	72
20	gTeX-References Implementation	78
20.1	Document URIs and URLs	78
20.2	Setting Reference Targets	80
20.3	Using References	81
21	gTeX-Modules Implementation	83
21.1	The module environment	86
21.2	Invoking modules	92
22	gTeX-Module Inheritance Implementation	94
22.1	SMS Mode	94
22.2	Inheritance	98

23	sTeX-Symbols Implementation	103
23.1	Symbol Declarations	103
23.2	Notations	109
24	sTeX-Terms Implementation	118
24.1	Symbol Invocations	118
24.2	Terms	121
24.3	Notation Components	127
25	sTeX-Structural Features Implementation	130
25.1	Imports with modification	130
25.2	The feature environment	131
25.3	Features	132
26	sTeX-Statements Implementation	138
26.1	Definitions	138
26.2	Assertions	141
26.3	Examples	143
26.4	Logical Paragraphs	145
27	The Implementation	148
27.1	Package Options	148
27.2	Proofs	148
27.3	Justifications	154
28	sTeX-Others Implementation	156
29	sTeX-Metatheory Implementation	157
30	Tikzinput Implementation	160
31	document-structure.sty Implementation	162
31.1	The OMDoc Class	162
31.2	Class Options	162
31.3	Beefing up the document environment	163
31.4	Implementation: OMDoc Package	163
31.5	Package Options	163
31.6	Document Structure	165
31.7	Front and Backmatter	168
31.8	Global Variables	170
32	MiKoSlides – Implementation	171
32.1	Class and Package Options	171
32.2	Notes and Slides	173
32.3	Header and Footer Lines	177
32.4	Frame Images	178
32.5	Colors and Highlighting	179
32.6	Sectioning	180
32.7	Excursions	182

33 The Implementation	184
33.1 Package Options	184
33.2 Problems and Solutions	185
33.3 Multiple Choice Blocks	190
33.4 Including Problems	191
33.5 Reporting Metadata	192
34 Implementation: The hwexam Class	194
34.1 Class Options	194
35 Implementation: The hwexam Package	196
35.1 Package Options	196
35.2 Assignments	197
35.3 Including Assignments	200
35.4 Typesetting Exams	201
35.5 Leftovers	203

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:N</code>	\underline{TF} \star

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

<code>\c_stex_pwd_seq</code>	Store the current working directory as path-sequence and string, respectively, and the (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 .sms-file.
<hr/> <hr/> <code>\stex_in_repository:nn</code>	<code>\stex_in_repository:nn{<repository-name>}{<code>}</code> Change the current repository to <code>{<repository-name>}</code> (or not, if <code>{<repository-name>}</code> is empty), and passes its ID on to <code>{<code>}</code> as #1. Switches back to the previous repository after executing <code>{<code>}</code> .
<hr/> <hr/> <code>\mhpath *</code>	<code>\mhpath{<archive-ID>}{<filename>}</code> Expands to the full path of file <code><filename></code> in repository <code><archive-ID></code> . Does not check whether the file or the repository exist.
<hr/> <hr/> <code>\inputref</code> <hr/> <code>\inputref:nn</code>	<code>\inputref[<archive-ID>]{<filename>}</code> <code>\inputs</code> the file <code><filename></code> in repository <code><archive-ID></code> .
<hr/> <hr/> <code>\libinput</code>	<code>\libinput{<filename>}</code> Inputs <code><filename>.tex</code> from the <code>lib</code> folders in the current archive and the <code>meta-inf</code> -archive of the current archive group (if existent). Throws an error if no file by that name exists in either folder, includes both if both exist.

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_str`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

5.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

Test 5

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

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

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

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

`\stex_invoke_module:n`

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

Test 6

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




`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

`\stex_in_smsmode:nn`

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

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

Test 7

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

6.1.2 Imports and Inheritance

`\importmodule`

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

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

Test 8

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

Module 6.1.1[Foo]

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

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

Module 6.1.2[Importtest]

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

Module 6.1.3[Importtest2]

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

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

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

Test 9

```

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

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

```

Module 6.1.4[UseTest1]

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

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

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

Test 10

```

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

```

Circular dependencies:

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

`\stex_import_module_uri:nn`

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

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

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

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

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

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

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

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

Chapter 7

STEX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

<u><code>\symdecl</code></u>	<code>\symdecl[⟨args⟩]{⟨macroname⟩}</code>
------------------------------	--

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

- **name**: An (OMDOC) name. By default equal to `⟨macroname⟩`.
- **type**: An (ideally semantic) term. Not used by STEX, but passed on to MMT for semantic services.
- **local**: A boolean (by default false). If set, this declaration will not be added to the module content, i.e. importing the current module will not make this declaration available.
- **args**: Specifies the “signature” of the semantic macro. Can be either an integer $0 \leq n \leq 9$, or a (more precise) sequence of the following characters:
 - i a “normal” argument, e.g. `\symdecl[args=ii]{plus}` allows for `\plus{2}{2}`.
 - a an *associative* argument; i.e. a sequence of arbitrarily many arguments provided as a comma-separated list, e.g. `\symdecl[args=a]{plus}` allows for `\plus{2,2,2}`.
 - b a *variable* argument. Is treated by STEX like an i-argument, but an application is turned into an OMBind in OMDoc, binding the provided variable in the subsequent arguments of the operator; e.g. `\symdecl[args=bi]{forall}` allows for `\forall{x\in\Nat}{x\geq0}`.

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

`\stex_notation_do:nn`

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

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

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

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

Test 12

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

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

`\symdef`

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

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

Test 13

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

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

Chapter 8

STEX-Terms

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

8.1 Macros and Environments

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

<hr/> <hr/>	$\backslash\infprec$ \backslashneginfprec	Maximal and minimal notation precedences.
<hr/> <hr/>	\backslashdobrackets	$\backslashdobrackets \{ \langle body \rangle \}$ Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S I E X brackets (by default (and)), which can be changed temporarily using \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  $\bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
 $\displaystyle \bar{plus}\{a,\mult\{b,c\}\}$  and  $\displaystyle \bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$ 
\withbrackets[ $\{ \}$ ]{  $\displaystyle \bar{plus}\{a,\mult\{b,c\}\}$  and  $\displaystyle \bar{mult}\{a,\plus\{\frac{ab}{c}\}\}$  }
\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

ST_EX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

`mathstructure` TODO

Chapter 10

TeX-Statements

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

10.1 Macros and Environments

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

Chapter 11

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

11.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

We will go over the general intuition by way of our running example (see Figure 1 for the source and Figure 2 for the formatted result).⁴

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

11.2 The User Interface

11.2.1 Package Options

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

11.2.2 Proofs and Proof steps

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

`spfidea`

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

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

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

11.2.3 Justifications

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

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

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

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

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

11.2.4 Proof Structure

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

1. The numbering scheme of proofs cannot be changed. It is more geared for teaching proof structures (the author's main use case) and not for writing papers. reported by Tobias Pfeiffer (fixed)
2. currently proof steps are formatted by the L^AT_EX `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 $\S\text{TeX}$ collection, a version of $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ that allows to markup $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ documents semantically without leaving the document format, essentially turning $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ into a document format for mathematical knowledge management (MKM).

This package supplies an infrastructure for writing OMDOC documents in $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. This includes a simple structure sharing mechanism for $\S\text{TeX}$ 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 $\S\text{TeX}$ sources, or after translation.

14.1 Introduction

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

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

14.2.2 Document Structure

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

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

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

`blindomgroup` L^AT_EX automatically computes the sectioning level, from the nesting of `omgroup` environments. But sometimes, we want to skip levels (e.g. to use a subsection* as an introduction for a chapter). Therefore the `omdoc` package provides a variant `blindomgroup`

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

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

that does not produce markup, but increments the sectioning level and logically groups document parts that belong together, but where traditional document markup relies on convention rather than explicit markup. The `blindomgroup` environment is useful e.g. for creating frontmatter at the correct level. Example 3 shows a typical setup for the outer document structure of a book with parts and chapters. We use two levels of `blindomgroup`:

- The outer one groups the introductory parts of the book (which we assume to have a sectioning hierarchy topping at the part level). This `blindomgroup` makes sure that the introductory remarks become a “chapter” instead of a “part”.
- The inner one groups the frontmatter³ and makes the preface of the book a section-level construct. Note that here the `display=flow` on the `omgroup` environment prevents numbering as is traditional for prefaces.

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`

`\CurrentSectionLevel`

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

14.2.3 Ignoring Inputs

`ignore`
`showignores`

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

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

is given to the `omdoc` class or `package`. But in the generated OMDoc result, the body is marked up with a `ignore` element. This is useful in two situations. For

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

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

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

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

14.2.4 Structure Sharing

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

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.⁷

14.2.5 Global Variables

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

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

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

14.2.6 Colors

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

14.3 Limitations

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

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

Chapter 15

Slides and Course Notes

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

15.1 Introduction

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

15.2 The User Interface

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

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

15.2.1 Package Options

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

- | | |
|---|--|
| <code>slides</code>
<code>notes</code> | <ul style="list-style-type: none">• The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see Section 15.2.2). |
| <code>sectocframes</code> | <ul style="list-style-type: none">• If the option <code>sectocframes</code> is given, then for the <code>omgroups</code>, special frames with the <code>omgroup</code> title (and number) are generated. |

<code>showmeta</code>	<ul style="list-style-type: none"> • <code>showmeta</code>. If this is set, then the metadata keys are shown (see [Koh20b] for details and customization options).
<code>frameimages</code> <code>fiboxed</code>	<ul style="list-style-type: none"> • If the option <code>frameimages</code> is set, then slide mode also shows the <code>\frameimage</code>-generated frames (see section 15.2.4). If also the <code>fiboxed</code> option is given, the slides are surrounded by a box.
<code>topsect</code>	<ul style="list-style-type: none"> • <code>topsect=<sect></code> can be used to specify the top-level sectioning level; the default for <code><sect></code> is <code>section</code>.

15.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

`\ifnotes` Note the use of the `\ifnotes` conditional, which allows different treatment between `notes` and `slides` mode – manually setting `\notesttrue` or `\notesfalse` is strongly discouraged however.

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

⁴MK: it would be very nice, if we did not need this environment, and this should be possible in principle, but not without intensive L^AT_EX trickery. Hints to the author are welcome.

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

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

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

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

15.2.3 Header and Footer Lines of the Slides

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

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

15.2.4 Frame Images

Sometimes, we want to integrate slides as images after all – e.g. because we already have a PowerPoint presentation, to which we want to add \LaTeX notes. In this case we can use `\frameimage[<opt>]{<path>}`, where `<opt>` are the options of `\includegraphics` from the `graphicx` package [CR99] and `<path>` is the file path (extension can be left off like in `\includegraphics`). We have added the `label` key that allows to give a frame label that can be referenced like a regular `beamer` frame.⁹

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

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

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

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


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

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

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

`\mhframeimage{baz/foobar}`

15.2.5 Colors and Highlighting

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

15.2.6 Front Matter, Titles, etc.

15.2.7 Excursions

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

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

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

```
\activateexcursion where \activateexcursion{<path>} augments the \printexcursions macro by a
\printexcursions call \inputref{<path>}. In this way, 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 `solutions`⁵. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

16.2 The User Interface

16.2.1 Package Options

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

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

16.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

```
Problem0.0 ()
How many Elephants can you fit into a Volkswagen beetle?


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

16.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

16.2.4 Including Problems

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

16.2.5 Reporting Metadata

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

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

16.3 Limitations

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

1. none reported yet

```

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

```

Problem0.0 ()

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

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

Problem0.0 ()

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

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

Example 7: A Problem with a multiple choice block

Chapter 17

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

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

Contents

17.1 Introduction

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

17.2 The User Interface

17.2.1 Package and Class Options

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

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

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

17.2.2 Assignments

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

17.2.3 Typesetting Exams

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

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

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

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

17.2.4 Including Assignments

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

17.3 Limitations

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

1. none reported yet.

Name:
MatriculationNumber:

2022-01-27

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
23 %\RequirePackage{morewrites}
24 %\RequirePackage{amsmath}
25
```

Package options:

```

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

```

\stex The ST_EX logo:

\sTeX

```

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

```

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

18.3 Messages and logging

```

43 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

18.4 Persistence

78 `<@@=stex_persist>`

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

```

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

```

(End definition for `\c__stex_persist_sms_iow`.)

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

```

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

```

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

18.5 HTML Annotations

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

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

```

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

```

`\if@latexml` Conditionals for L^AT_EX_ML:

```

\latexml_if_p:
\latexml_if:TF
100 \ifcsname if@latexml\endcsname\else

```

```

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

```

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

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

```

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

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

`_stex_annotate_checkempty:n`

```

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

```

(End definition for _stex_annotate_checkempty:n.)

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

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EXML, 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_EXML-implementations resort to perl bindings.

```

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

```

```

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

```

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

18.6 Languages

```

234 <@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

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

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

```

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

```

18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```


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

\stex_reactivate_macro:N

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

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

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

Chapter 19

STEX -MathHub Implementation

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

19.1 Generic Path Handling

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

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

```

```

308     }
309     \seq_set_eq:NN #1 \l_tmpa_tl
310   }
311   \stex_path_canonicalize:N #1
312 }
313 }
314 \cs_generate_variant:Nn \stex_path_from_string:Nn
315 { NV, cn, cV }

```

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

```

\stex_path_to_string:NN
\stex_path_to_string:N
316 \cs_new_protected:Nn \stex_path_to_string:NN {
317   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
318 }
319
320 \cs_new:Nn \stex_path_to_string:N {
321   \seq_use:Nn #1 /
322 }

```

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

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
323 \str_const:Nn \c__stex_path_dot_str {.}
324 \str_const:Nn \c__stex_path_up_str {...}

```

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

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

```

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

```

```

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

```

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

`\stex_path_if_absolute_p:N`
`\stex_path_if_absolute:N \underline{TF}`

```

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

```

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

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

(End definition for `\c_stex_pwd_seq` and `\c_stex_pwd_str`. These variables are documented on page 11.)

19.3 File Hooks and Tracking

387 <@@=stex_files>

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

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

`\g__stex_files_stack` keeps track of file changes

388 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

390 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

391 `\c_stex_mainfile_str`

(End definition for `\c_stex_mainfile_seq` and `\c_stex_mainfile_str`. These variables are documented on page 11.)

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

```

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

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

19.4 MathHub Repositories

```

416 <@@=stex_mathhub>

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

```

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

```

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

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
453 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

489 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

`_stex_mathhub_parse_manifest:n` Stores the entries in manifest file in the corresponding property list:

```

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

```

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```



```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

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

```

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

```

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

```

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

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

```

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

\mhpath

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

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

\libinput

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

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

```
680 </package>
```

Chapter 20

STEX -References Implementation

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

20.1 Document URIs and URLs

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

```

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

```

```

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

```

20.2 Setting Reference Targets

```

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

```

```

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

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

```

20.3 Using References

```

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

```

```

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

```


Chapter 21

STEX -Modules Implementation

```
897 <*package>
898
899 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
900
901 <@@=stex_modules>
    Warnings and error messages
902 \msg_new:nnn{stex}{error/unknownmodule}{
903   No~module~#1~found
904 }
905 \msg_new:nnn{stex}{error/syntax}{
906   Syntax~error:~#1
907 }
908 \msg_new:nnn{stex}{error/siglanguage}{
909   Module~#1~declares~signature~#2,~but~does~not~
910   declare~its~language
911 }
912
913 \msg_new:nnn{stex}{error/conclictingmodules}{
914   Comflicting~imports~for~module~#1
915 }

\l_stex_current_module_str The current module:
916 \str_new:N \l_stex_current_module_str
    (End definition for \l_stex_current_module_str. This variable is documented on page 15.)

\l_stex_all_modules_seq Stores all available modules
917 \seq_new:N \l_stex_all_modules_seq
    (End definition for \l_stex_all_modules_seq. This variable is documented on page 15.)

\stex_if_in_module_p:
\stex_if_in_module:TF
918 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
919   \str_if_empty:NTF \l_stex_current_module_str
920     \prg_return_false: \prg_return_true:
921 }
```

(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`

```
922 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
923   \prop_if_exist:cTF { c_stex_module_#1_prop }
924   \prg_return_true: \prg_return_false:
925 }
```

(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:

```
926 \cs_new_protected:Nn \stex_add_to_current_module:n {
927   \tl_gput_right:cn {c_stex_module_\l_stex_current_module_str _code} { #1 }
928 }
929 \cs_new_protected:Npn \STEXexport {
930   \beginngroup
931   \newlinechar=-1\relax
932   \endlinechar=-1\relax
933   %\catcode'\ = 9\relax
934   \expandafter\endgroup\STEXexport:n
935 }
936 \cs_new_protected:Nn \STEXexport:n {
937   \ignorespaces #1
938   \stex_add_to_current_module:n { \ignorespaces #1 }
939   \stex_smsmode_set_codes:
940 }
941 \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`

```
942 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
943   \str_set:Nx \l_tmpa_str { #1 }
944   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _constants} { \l_tmpa_str }
945 }
946
947 \cs_new_protected:Nn \stex_add_field_to_current_module:n {
948   \str_set:Nx \l_tmpa_str { #1 }
949   \seq_gput_right:co {c_stex_module_\l_stex_current_module_str _fields} { \l_tmpa_str }
950 }
```

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

`\stex_collect_imports:nn`

```
951 \cs_new_protected:Nn \stex_collect_imports:nn {
952   \seq_clear:N \l_stex_collect_imports_seq
953   \__stex_modules_collect_imports:n {#1}
954 }
955 \cs_new_protected:Nn \__stex_modules_collect_imports:n {
956   \seq_map_inline:cn {c_stex_module_#1_imports} {
957     \seq_if_in:NnF \l_stex_collect_imports_seq { ##1 } {
958       \__stex_modules_collect_imports:n { ##1 }
959     }
960 }
```

```

960 }
961 \seq_if_in:NnF \l_stex_collect_imports_seq { #1 } {
962   \seq_put_right:Nn \l_stex_collect_imports_seq { #1 }
963 }
964 }

```

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

`\stex_add_import_to_current_module:n`

```

965 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
966   \str_set:Nx \l_tmpa_str { #1 }
967   \exp_args:Nno
968   \seq_if_in:cnF{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str{
969     \seq_gput_right:co{c_stex_module\l_stex_current_module_str_imports}\l_tmpa_str
970   }
971 }

```

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

`\stex_modules_compute_namespace:nN`

Computes the appropriate namespace from the top-level namespace of a repository (#1) and a file path (#2).

```

972 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
973   \str_set:Nx \l_tmpa_str { #1 }
974   \seq_set_eq:NN \l_tmpa_seq #2
975   % split off file extension
976   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
977   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
978   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
979   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
980
981   \bool_set_true:N \l_tmpa_bool
982   \bool_while_do:Nn \l_tmpa_bool {
983     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
984     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
985       {source} { \bool_set_false:N \l_tmpa_bool }
986     }{}{
987       \seq_if_empty:NT \l_tmpa_seq {
988         \bool_set_false:N \l_tmpa_bool
989       }
990     }
991   }
992
993   \stex_path_to_string:NN \l_tmpa_seq \l_stex_modules_subpath_str
994   \str_if_empty:NTF \l_stex_modules_subpath_str {
995     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
996   }{
997     \str_set:Nx \l_stex_modules_ns_str {
998       \l_tmpa_str/\l_stex_modules_subpath_str
999     }
1000   }
1001 }

```

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

Stores its return values in:

```

\l_stex_modules_ns_str
\l_stex_modules_subpath_str
1002 \str_new:N \l_stex_modules_ns_str
1003 \str_new:N \l_stex_modules_subpath_str

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

```

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

```

1004 \cs_new_protected:Nn \stex_modules_current_namespace: {
1005   \str_clear:N \l_stex_modules_subpath_str
1006   \prop_get:NnTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
1007     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
1008   }{
1009     % split off file extension
1010     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1011     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
1012     \exp_args:Nnno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
1013     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
1014     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
1015     \str_set:Nx \l_stex_modules_ns_str {
1016       file:/\stex_path_to_string:N \l_tmpa_seq
1017     }
1018   }
1019 }

```

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

21.1 The module environment

module arguments:

```

1020 \keys_define:nn { stex / module } {
1021   title      .str_set_x:N = \l_stex_module_title_str ,
1022   ns         .str_set_x:N = \l_stex_module_ns_str ,
1023   lang       .str_set_x:N = \l_stex_module_lang_str ,
1024   sig        .str_set_x:N = \l_stex_module_sig_str ,
1025   creators   .str_set_x:N = \l_stex_module_creators_str ,
1026   contributors .str_set_x:N = \l_stex_module_contributors_str ,
1027   meta       .str_set_x:N = \l_stex_module_meta_str ,
1028   srccite    .str_set_x:N = \l_stex_module_srccite_str
1029 }
1030
1031 \cs_new_protected:Nn \__stex_modules_args:n {
1032   \str_clear:N \l_stex_module_title_str
1033   \str_clear:N \l_stex_module_ns_str
1034   \str_clear:N \l_stex_module_lang_str
1035   \str_clear:N \l_stex_module_sig_str
1036   \str_clear:N \l_stex_module_creators_str
1037   \str_clear:N \l_stex_module_contributors_str
1038   \str_clear:N \l_stex_module_meta_str
1039   \str_clear:N \l_stex_module_srccite_str
1040   \keys_set:nn { stex / module } { #1 }
1041 }

```

```

1042
1043 % module parameters here? In the body?
1044

```

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

```

1045 \cs_new_protected:Nn \stex_module_setup:nn {
1046   \str_set:Nx \l_stex_module_name_str { #2 }
1047   \__stex_modules_args:n { #1 }

```

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

```

1048   \stex_if_in_module:TF {
1049     % Nested module
1050     \prop_get:cnN {c_stex_module_\l_stex_current_module_str _prop}
1051       { ns } \l_stex_module_ns_str
1052     \str_set:Nx \l_stex_module_name_str {
1053       \prop_item:cn {c_stex_module_\l_stex_current_module_str _prop}
1054         { name } / \l_stex_module_name_str
1055     }
1056   }{
1057     % not nested:
1058     \str_if_empty:NT \l_stex_module_ns_str {
1059       \stex_modules_current_namespace:
1060       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
1061       \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
1062         / {\l_stex_module_ns_str}
1063       \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1064       \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
1065         \str_set:Nx \l_stex_module_ns_str {
1066           \stex_path_to_string:N \l_tmpa_seq
1067         }
1068       }
1069     }
1070   }

```

Next, we determine the language of the module:

```

1071   \str_if_empty:NT \l_stex_module_lang_str {
1072     \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
1073     \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
1074     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
1075     \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
1076     \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
1077       \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
1078         inferred~from~file~name}
1079       \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
1080     }
1081   }
1082
1083   \str_if_empty:NF \l_stex_module_lang_str {
1084     \prop_get:NVNTF {c_stex_languages_prop \l_stex_module_lang_str
1085       \l_tmpa_str {
1086       \ltx@ifpackageloaded{babel}{
1087         \exp_args:Nx \selectlanguage { \l_tmpa_str }
1088       }{}

```

```

1089     } {
1090       \msg_error:nnx{stex}{error/unknownlanguage}{\l_tmpa_str}
1091     }
1092   }

```

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

```

1093   \str_if_empty:NTF \l_stex_module_sig_str {
1094     \exp_args:Nnx \prop_gset_from_keyval:cn {
1095       c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _prop
1096     } {
1097       name      = \l_stex_module_name_str ,
1098       ns        = \l_stex_module_ns_str ,
1099       file       = \exp_not:o { \g_stex_currentfile_seq } ,
1100       lang       = \l_stex_module_lang_str ,
1101       sig        = \l_stex_module_sig_str ,
1102       meta       = \l_stex_module_meta_str
1103     }
1104     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _imports}
1105     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _fields}
1106     \seq_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _constants}
1107     \tl_clear:c {c_stex_module_\l_stex_module_ns_str?\l_stex_module_name_str _code}
1108     \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}

```

We load the metatheory:

```

1109   \str_if_empty:NT \l_stex_module_meta_str {
1110     \str_set:Nx \l_stex_module_meta_str {
1111       \c_stex_metatheory_ns_str ? Metatheory
1112     }
1113   }
1114   \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1115     \bool_set_true:N \l_stex_in_meta_bool
1116     \exp_args:Nx \stex_add_to_current_module:n {
1117       \bool_set_true:N \l_stex_in_meta_bool
1118       \stex_activate_module:n {\l_stex_module_meta_str}
1119       \bool_set_false:N \l_stex_in_meta_bool
1120     }
1121     \stex_activate_module:n {\l_stex_module_meta_str}
1122     \bool_set_false:N \l_stex_in_meta_bool
1123   }
1124 }{
1125   \str_if_empty:NT \l_stex_module_lang_str {
1126     \msg_error:nnxx{stex}{error/siglanguage}{
1127       \l_stex_module_ns_str?\l_stex_module_name_str
1128     }{\l_stex_module_sig_str}
1129   }
1130
1131   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1132   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
1133   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
1134   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
1135   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
1136   \str_set:Nx \l_tmpa_str {
1137     \stex_path_to_string:N \l_tmpa_seq /

```

```

1138     \l_tmpa_str . \l_stex_module_sig_str .tex
1139 }
1140 \IfFileExists \l_tmpa_str {
1141     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1142         \seq_clear:N \l_stex_all_modules_seq
1143         %\prop_clear:N \l_stex_current_module_prop
1144         \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1145         \input { \l_tmpa_str }
1146     }
1147 }{
1148     \msg_error:nnx{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1149 }
1150 \stex_activate_module:n {
1151     \l_stex_module_ns_str ? \l_stex_module_name_str
1152 }
1153 %\prop_set_eq:Nc \l_stex_current_module_prop {
1154 %   c_stex_module_
1155 %   \l_stex_module_ns_str ?
1156 %   \l_stex_module_name_str
1157 %   _prop
1158 %}
1159 \str_set:Nx\l_stex_current_module_str{\l_stex_module_ns_str?\l_stex_module_name_str}
1160 }
1161 }

```

(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}

1162 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1163     \stex_reactivate_macro:N \STEXexport
1164     \stex_reactivate_macro:N \importmodule
1165     \stex_reactivate_macro:N \symdecl
1166     \stex_reactivate_macro:N \notation
1167     \stex_reactivate_macro:N \symdef
1168     \stex_module_setup:nn{#1}{#2}
1169 }
1170 \stex_debug:nn{modules}{
1171     New~module:\\
1172     Namespace:~\l_stex_module_ns_str\\
1173     Name:~\l_stex_module_name_str\\
1174     Language:~\l_stex_module_lang_str\\
1175     Signature:~\l_stex_module_sig_str\\
1176     Metatheory:~\l_stex_module_meta_str\\
1177     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1178 }
1179 }
1180 \seq_put_right:Nx \l_stex_all_modules_seq {
1181     \l_stex_module_ns_str ? \l_stex_module_name_str
1182 }
1183 }
1184 % \seq_gput_right:Nx \g_stex_modules_in_file_seq
1185 %     { \l_stex_module_ns_str ? \l_stex_module_name_str }

```

```

1186
1187 \stex_if_smsmode:TF {
1188   \stex_smsmode_set_codes:
1189 } {
1190   \begin{stex_annotate_env} {theory} {
1191     \l_stex_module_ns_str ? \l_stex_module_name_str
1192   }
1193
1194   \stex_annotate_invisible:nnn{header}{} {
1195     \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1196     \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1197     \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1198       \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1199     }
1200   }
1201 }
1202 % TODO: Inherit metatheory for nested modules?
1203 }
1204 \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}`

```

1205 \cs_new_protected:Nn \_stex_modules_end_module: {
1206 % \str_set:Nx \l_tmpa_str {
1207 %   c_stex_module_
1208 %   \prop_item:Nn \l_stex_current_module_prop { ns } ?
1209 %   \prop_item:Nn \l_stex_current_module_prop { name }
1210 %   _prop
1211 % }
1212 %^^A \prop_new:c { \l_tmpa_str }
1213 % \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
1214 \stex_debug:nn{modules}{Closing module~\prop_item:cn {c_stex_module_\l_stex_current_module_
1215 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1216 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1217 \NewDocumentEnvironment { @module } { 0{} m } {
1218   \par
1219   \_stex_modules_begin_module:nn{#1}{#2}
1220 } {
1221   \_stex_modules_end_module:
1222   \stex_if_smsmode:TF {
1223 %     \exp_args:Nx \stex_add_to_sms:n {
1224 %       \prop_gset_from_keyval:cn {
1225 %         c_stex_module_
1226 %         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1227 %         \prop_item:Nn \l_stex_current_module_prop { name }
1228 %         _prop
1229 %       } {
1230 %         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1231 %         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,

```



```

1232 %         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1233 %         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1234 %         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1235 %         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1236 %     }
1237 % }
1238 }{
1239   \end{stex_annotate_env}
1240 }
1241 }

```

\stex_modules_heading: Code for document headers

```

1242 \cs_if_exist:NTF \thesection {
1243   \newcounter{module}[section]
1244 }{
1245   \newcounter{module}
1246 }
1247
1248 \bool_if:NT \c_stex_showmods_bool {
1249   \latexml_if:F { \RequirePackage{mdframed} }
1250 }
1251
1252 \cs_new_protected:Nn \stex_modules_heading: {
1253   \stepcounter{module}
1254   \par
1255   \bool_if:NT \c_stex_showmods_bool {
1256     \noindent{\textbf{Module} ~
1257       \cs_if_exist:NT \thesection {\thesection.}
1258       \themodule ~ [\l_stex_module_name_str]
1259     }
1260     \str_if_empty:NTF \l_stex_module_title_str {
1261       }{
1262         \quad(\l_stex_module_title_str)\hfill
1263       }\par
1264     }
1265     \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1266     % TODO
1267     \stex_ref_new_doc_target:n \l_stex_module_name_str
1268   }

```

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

Finally:

```

1269 \NewDocumentEnvironment { module } { 0{} m } {
1270   \bool_if:NT \c_stex_showmods_bool {
1271     \begin{mdframed}
1272   }
1273   \begin{@module}[#1]{#2}
1274     \stex_modules_heading:
1275   }{
1276     \end{@module}
1277     \bool_if:NT \c_stex_showmods_bool {
1278       \end{mdframed}
1279     }
1280   }

```

21.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n 1281 \NewDocumentCommand \STEXModule { m } {
1282   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1283   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1284   \tl_set:Nn \l_tmpa_tl {
1285     \msg_error:nnx{stex}{error/unknownmodule}{#1}
1286   }
1287   \seq_map_inline:Nn \l_stex_all_modules_seq {
1288     \str_set:Nn \l_tmpb_str { ##1 }
1289     \str_if_eq:eeT { \l_tmpa_str } {
1290       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1291     } {
1292       \seq_map_break:n {
1293         \tl_set:Nn \l_tmpa_tl {
1294           \stex_invoke_module:n { ##1 }
1295         }
1296       }
1297     }
1298   }
1299   \l_tmpa_tl
1300 }
1301
1302 \cs_new_protected:Nn \stex_invoke_module:n {
1303   \stex_debug:nn{modules}{Invoking~module~#1}
1304   \peek_charcode_remove:NTF ! {
1305     \__stex_modules_invoke_uri:nN { #1 }
1306   } {
1307     \peek_charcode_remove:NTF ? {
1308       \__stex_modules_invoke_symbol:nn { #1 }
1309     } {
1310       \msg_error:nnx{stex}{error/syntax}{
1311         ?~or~!~expected~after~
1312         \c_backslash_str STEXModule{#1}
1313       }
1314     }
1315   }
1316 }
1317
1318 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1319   \str_set:Nn #2 { #1 }
1320 }
1321
1322 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1323   \stex_invoke_symbol:n{#1?#2}
1324 }

```

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

```

\stex_activate_module:n
1325 \bool_new:N \l_stex_in_meta_bool
1326 \bool_set_false:N \l_stex_in_meta_bool

```

```

1327 \cs_new_protected:Nn \stex_activate_module:n {
1328   \stex_debug:nn{modules}{Activating~module~#1}
1329   \seq_if_in:NnT \l_stex_implicit_morphisms_seq { #1 }{
1330     \msg_error:nnn{stex}{error/conclictingmodules}{ #1 }
1331   }
1332   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1333     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1334     \use:c{ c_stex_module_#1_code }
1335   }
1336 }

```

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

```

1337 </package>

```

Chapter 22

sTeX -Module Inheritance Implementation

```
1338 <*package>
1339
1340 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1341
```

22.1 SMS Mode

```
1342 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1343 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1344 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1345 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1346
1347 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1348   \makeatletter
1349   \makeatother
1350   \ExplSyntaxOn
1351   \ExplSyntaxOff
1352 }
1353
1354 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1355   \symdef
1356   \importmodule
1357   \notation
1358   \symdecl
1359   \STEXexport
1360 }
1361
1362 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1363   \tl_to_str:n {
1364     module,
1365     @module
```

```

1366 }
1367 }

```

(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

```

```

1368 \bool_new:N \g__stex_smsmode_bool
1369 \bool_set_false:N \g__stex_smsmode_bool
1370 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1371   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1372 }

```

(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

```

```

1373 \bool_new:N \g__stex_smsmode_catcode_bool
1374 \bool_set_false:N \g__stex_smsmode_catcode_bool
1375 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1376   \bool_if:NTF \g__stex_smsmode_catcode_bool
1377   \prg_return_true: \prg_return_false:
1378 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1379 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1380   \stex_if_smsmode:T {
1381     \__stex_smsmode_if_catcodes:F {
1382       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1383       \exp_after:wN \char_gset_active_eq:NN
1384       \c_backslash_str \__stex_smsmode_cs:
1385       \tex_global:D \char_set_catcode_active:N \
1386       \tex_global:D \char_set_catcode_other:N $
1387       \tex_global:D \char_set_catcode_other:N ^
1388       \tex_global:D \char_set_catcode_other:N _
1389       \tex_global:D \char_set_catcode_other:N &
1390       \tex_global:D \char_set_catcode_other:N ##
1391     }
1392   }
1393 } \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.

```

1394 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1395   \__stex_smsmode_if_catcodes:T {
1396     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1397     \exp_after:wN \tex_global:D \exp_after:wN
1398     \char_set_catcode_escape:N \c_backslash_str
1399     \tex_global:D \char_set_catcode_math_toggle:N $
1400     \tex_global:D \char_set_catcode_math_superscript:N ^
1401     \tex_global:D \char_set_catcode_math_subscript:N _
1402     \tex_global:D \char_set_catcode_alignment:N &
1403     \tex_global:D \char_set_catcode_parameter:N ##
1404   }
1405 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1406 \cs_new_protected:Nn \stex_in_smsmode:nn {
1407   \vbox_set:Nn \l_tmpa_box {
1408     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1409     \bool_gset_true:N \g__stex_smsmode_bool
1410     \stex_smsmode_set_codes:
1411     #2
1412     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1413     \stex_if_smsmode:F {
1414       \__stex_smsmode_unset_codes:
1415     }
1416   }
1417   \box_clear:N \l_tmpa_box
1418 }

```

(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:

```

1419 \cs_new_protected:Nn \_stex_smsmode_cs: {
1420   \str_clear:N \l_tmpa_str
1421   \peek_analysis_map_inline:n {
1422     % #1: token (one expansion)
1423     % #2: charcode
1424     % #3 catcode
1425     \token_if_eq_charcode:NNTF ##3 B {
1426       % token is a letter
1427       \exp_args:NN \str_put_right:Nn \l_tmpa_str { ##1 }
1428     } {
1429       \str_if_empty:NTF \l_tmpa_str {
1430         % we don't allow (or need) single non-letter CSs
1431         % for now
1432         \peek_analysis_map_break:
1433       }{
1434         \str_if_eq:onTF \l_tmpa_str { begin } {
1435           \peek_analysis_map_break:n {
1436             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1437           }
1438         } {
1439           \str_if_eq:onTF \l_tmpa_str { end } {
1440             \peek_analysis_map_break:n {
1441               \exp_after:wN \_stex_smsmode_checkend:n ##1
1442             }
1443           } {
1444             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1445             \exp_args:NNNo \exp_args:NN \tl_if_in:NnTF
1446             \g_stex_smsmode_allowedmacros_tl
1447             { \use:c{\l_tmpa_str} } {
1448               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1449               \peek_analysis_map_break:n {
1450                 \exp_after:wN \l_tmpa_tl ##1
1451               }

```

```

1452     } {
1453         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1454         \g_stex_smsmode_allowedmacros_escape_tl
1455         { \use:c{\l_tmpa_str} } {
1456             \__stex_smsmode_unset_codes:
1457             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1458             % TODO \__stex_smsmode_rescan_cs:
1459             % \int_compare:nNnTF {##2} = {92} {
1460             %     \peek_analysis_map_break:n {
1461             %         \__stex_smsmode_unset_codes:
1462             %         \__stex_smsmode_rescan_cs:
1463             %     }
1464             % } {
1465             %     \peek_analysis_map_break:n {
1466             %         \exp_after:wN \l_tmpa_tl ##1
1467             %     }
1468             % }
1469             } {
1470                 \int_compare:nNnTF {##2} = {92} {
1471                     \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1472                 }{
1473                     \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1474                 }
1475             }
1476         }
1477     }
1478 }
1479 }
1480 }
1481 }
1482 }

```

(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:

```

1483 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1484     \str_clear:N \l_tmpb_str
1485     \peek_analysis_map_inline:n {
1486         \token_if_eq_charcode:NNTF ##3 B {
1487             % token is a letter
1488             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1489         } {
1490             \peek_analysis_map_break:n {
1491                 \exp_after:wN \use:c \exp_after:wN {
1492                     \exp_after:wN \l_tmpa_str\exp_after:wN
1493                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1494             }
1495         }
1496     }
1497 }

```

(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.

```

1498 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1499   \str_set:Nn \l_tmpa_str { #1 }
1500   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1501     \__stex_smsmode_unset_codes:
1502     \begin{#1}
1503   }
1504 }
```

(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.

```

1505 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1506   \str_set:Nn \l_tmpa_str { #1 }
1507   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1508     \end{#1}
1509   }
1510 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

```

1511 <@@=stex_importmodule>
```

`\stex_import_module_uri:nn`

```

1512 \cs_new_protected:Nn \stex_import_module_uri:nn {
1513   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1514   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1515
1516   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1517   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1518   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1519
1520   \stex_modules_current_namespace:
1521   \bool_lazy_all:nTF {
1522     {\str_if_empty_p:N \l__stex_importmodule_archive_str}
1523     {\str_if_empty_p:N \l__stex_importmodule_path_str}
1524     {\stex_if_module_exists_p:n { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } }
1525   }{
1526     \str_set_eq:NN \l__stex_importmodule_path_str \l_stex_modules_subpath_str
1527     \str_set_eq:NN \l_stex_module_ns
1528   }{
1529     \str_if_empty:NT \l__stex_importmodule_archive_str {
1530       \prop_if_empty:NF \l_stex_current_repository_prop {
1531         \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_s
1532       }
1533     }
1534     \str_if_empty:NTF \l__stex_importmodule_archive_str {
1535       \str_if_empty:NF \l__stex_importmodule_path_str {
1536         \str_set:Nx \l_stex_module_ns_str {
1537           \l_stex_module_ns_str / \l__stex_importmodule_path_str
1538         }
1539       }
```



```

1540   }{
1541     \stex_require_repository:n \l__stex_importmodule_archive_str
1542     \prop_get:cnN { c_stex_mathhub\_l__stex_importmodule_archive_str _manifest_prop } { ns
1543       \l_stex_module_ns_str
1544     \str_if_empty:NF \l__stex_importmodule_path_str {
1545       \str_set:Nx \l_stex_module_ns_str {
1546         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1547       }
1548     }
1549   }
1550 }
1551 }

```

(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>	1552 <code>\str_new:N \l__stex_importmodule_name_str</code>
<code>\l__stex_importmodule_path_str</code>	1553 <code>\str_new:N \l__stex_importmodule_archive_str</code>
<code>\l__stex_importmodule_file_str</code>	1554 <code>\str_new:N \l__stex_importmodule_path_str</code>
	1555 <code>\str_new:N \g__stex_importmodule_file_str</code>

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

```

\stex_import_require_module:nnnnn    {<ns>} {<archive-ID>} {<path>} {<name>}
1556 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1557   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1558
1559     % archive
1560     \str_set:Nx \l_tmpa_str { #2 }
1561     \str_if_empty:NTF \l_tmpa_str {
1562       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1563     } {
1564       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1565       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1566       \seq_put_right:Nn \l_tmpa_seq { source }
1567     }
1568
1569     % path
1570     \str_set:Nx \l_tmpb_str { #3 }
1571     \str_if_empty:NTF \l_tmpb_str {
1572       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1573
1574       \ltx@ifpackageloaded{babel} {
1575         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1576           { \language } \l_tmpb_str {
1577           \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1578         }
1579       } {
1580         \str_clear:N \l_tmpb_str
1581       }
1582
1583       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1584       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1585         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }

```

```

1586     }{
1587       \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1588       \IfFileExists{ \l_tmpa_str.tex }{
1589         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1590       }{
1591         % try english as default
1592         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1593         \IfFileExists{ \l_tmpa_str.en.tex }{
1594           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1595         }{
1596           \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1597         }
1598       }
1599     }
1600
1601   } {
1602     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1603     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1604
1605     \ltx@ifpackageloaded{babel} {
1606       \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1607         { \language } \l_tmpb_str {
1608         \msg_error:nnx{stex}{error/unknownlanguage}{\language}
1609       }
1610     } {
1611       \str_clear:N \l_tmpb_str
1612     }
1613
1614     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1615
1616     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1617     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1618       \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1619     }{
1620       \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1621       \IfFileExists{ \l_tmpa_str/#4.tex }{
1622         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1623       }{
1624         % try english as default
1625         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1626         \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1627           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1628         }{
1629           \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1630           \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1631             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1632           }{
1633             \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1634             \IfFileExists{ \l_tmpa_str.tex }{
1635               \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1636             }{
1637               % try english as default
1638               \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1639               \IfFileExists{ \l_tmpa_str.en.tex }{

```

```

1640         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1641     }{
1642         \msg_error:nnx{stex}{error/unknownmodule}{#1?#4}
1643     }
1644 }
1645 }
1646 }
1647 }
1648 }
1649 }
1650
1651 \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1652     \seq_clear:N \l_stex_all_modules_seq
1653     \str_clear:N \l_stex_current_module_str
1654     \str_set:Nx \l_tmpb_str { #2 }
1655     \str_if_empty:NF \l_tmpb_str {
1656         \stex_set_current_repository:n { #2 }
1657     }
1658     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1659     \input { \g__stex_importmodule_file_str }
1660 }
1661
1662 \stex_if_module_exists:nF { #1 ? #4 } {
1663     \msg_error:nnx{stex}{error/unknownmodule}{
1664         #1?#4~(in~file~\g__stex_importmodule_file_str)
1665     }
1666 }
1667 }
1668 \stex_activate_module:n { #1 ? #4 }
1669 }

```

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

`\importmodule`

```

1670 \NewDocumentCommand \importmodule { 0{} m } {
1671     \stex_import_module_uri:nn { #1 } { #2 }
1672     \stex_debug:nn{modules}{Importing~module:~
1673         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1674     }
1675     \stex_if_smsmode:F {
1676         \stex_import_require_module:nnnn
1677         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1678         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1679         \stex_annotate_invisible:nnn
1680         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1681     }
1682     \exp_args:Nx \stex_add_to_current_module:n {
1683         \stex_import_require_module:nnnn
1684         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1685         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1686     }
1687     \exp_args:Nx \stex_add_import_to_current_module:n {
1688         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1689     }

```

```

1690 \stex_smsmode_set_codes:
1691 }
1692 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1693 \NewDocumentCommand \usemodule { 0{} m } {
1694 \stex_if_smsmode:F {
1695 \stex_import_module_uri:nn { #1 } { #2 }
1696 \stex_import_require_module:nnnn
1697 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1698 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1699 \stex_annotate_invisible:nnn
1700 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1701 }
1702 \stex_smsmode_set_codes:
1703 }

```

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

```

1704 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1705 <*package>
1706
1707 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1708
```

Warnings and error messages

```
1709
```

23.1 Symbol Declarations

```
1710 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1711 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1717 \NewDocumentCommand \STEXsymbol { m } {
1718   \stex_get_symbol:n { #1 }
1719   \exp_args:No
1715   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1716 }
```

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

symdecl arguments:

```
1717 \keys_define:nn { stex / symdecl } {
1718   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1719   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1720   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1721   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1722   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1723   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1724   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1725   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1726 }
```

```

1727
1728 \bool_new:N \l_stex_symdecl_make_macro_bool
1729
1730 \cs_new_protected:Nn \__stex_symdecl_args:n {
1731   \str_clear:N \l_stex_symdecl_name_str
1732   \str_clear:N \l_stex_symdecl_args_str
1733   \bool_set_false:N \l_stex_symdecl_local_bool
1734   \tl_clear:N \l_stex_symdecl_type_tl
1735   \tl_clear:N \l_stex_symdecl_definiens_tl
1736
1737   \keys_set:nn { stex / symdecl } { #1 }
1738 }

```

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

```

1739
1740 \NewDocumentCommand \symdecl { s O{} m } {
1741   \__stex_symdecl_args:n { #2 }
1742   \IfBooleanTF #1 {
1743     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1744   } {
1745     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1746   }
1747   \stex_symdecl_do:n { #3 }
1748   \stex_smsmode_set_codes:
1749 }
1750 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1751 \cs_new_protected:Nn \stex_symdecl_do:n {
1752   \stex_if_in_module:F {
1753     % TODO throw error? some default namespace?
1754   }
1755
1756   \str_if_empty:NT \l_stex_symdecl_name_str {
1757     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1758   }
1759
1760   \prop_if_exist:cT { l_stex_symdecl_
1761     \l_stex_current_module_str ?
1762     \l_stex_symdecl_name_str
1763     _prop
1764   }{
1765     % TODO throw error (beware of circular dependencies)
1766   }
1767
1768   \prop_clear:N \l_tmpa_prop
1769   \prop_put:Nnx \l_tmpa_prop { module } { \l_stex_current_module_str }
1770   \seq_clear:N \l_tmpa_seq
1771   \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1772   \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1773   \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl

```

```

1774
1775 \exp_args:No \stex_add_constant_to_current_module:n {
1776   \l_stex_symdecl_name_str
1777 }
1778
1779 % arity/args
1780 \int_zero:N \l_tmpb_int
1781
1782 \bool_set_true:N \l_tmpa_bool
1783 \str_map_inline:Nn \l_stex_symdecl_args_str {
1784   \token_case_meaning:NnF ##1 {
1785     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1786     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1787     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1788     {\tl_to_str:n a} {
1789       \bool_set_false:N \l_tmpa_bool
1790       \int_incr:N \l_tmpb_int
1791     }
1792     {\tl_to_str:n B} {
1793       \bool_set_false:N \l_tmpa_bool
1794       \int_incr:N \l_tmpb_int
1795     }
1796   }{
1797     \msg_set:nnn{stex}{error/wrongargs}{
1798       args~value~in~symbol~declaration~for~
1799       \l_stex_current_module_str ?
1800       \l_stex_symdecl_name_str ~
1801       needs~to~be~
1802       i,~a,~b~or~B,~but~##1~given
1803     }
1804     \msg_error:nn{stex}{error/wrongargs}
1805   }
1806 }
1807 \bool_if:NTF \l_tmpa_bool {
1808   % possibly numeric
1809   \str_if_empty:NTF \l_stex_symdecl_args_str {
1810     \prop_put:Nnn \l_tmpa_prop { args } {}
1811     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1812   }{
1813     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1814     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1815     \str_clear:N \l_tmpa_str
1816     \int_step_inline:nn \l_tmpa_int {
1817       \str_put_right:Nn \l_tmpa_str i
1818     }
1819     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1820   }
1821 } {
1822   \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1823   \prop_put:Nnx \l_tmpa_prop { arity }
1824   { \str_count:N \l_stex_symdecl_args_str }
1825 }
1826 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1827

```

```

1828
1829 % semantic macro
1830
1831 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1832   \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1833     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1834   } }
1835
1836   \bool_if:NF \l_stex_symdecl_local_bool {
1837     \exp_args:Nx \stex_add_to_current_module:n {
1838       \tl_set:cn { #1 } { \stex_invoke_symbol:n {
1839         \l_stex_current_module_str ? \l_stex_symdecl_name_str
1840       } }
1841     }
1842   }
1843 }
1844
1845 % add to all symbols
1846
1847 \bool_if:NF \l_stex_symdecl_local_bool {
1848   \exp_args:Nx \stex_add_to_current_module:n {
1849     \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1850       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1851     }
1852   }
1853   \exp_args:Nx \stex_add_field_to_current_module:n {
1854     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1855   }
1856 }
1857
1858 \stex_debug:nn{symbols}{New~symbol:~
1859   \l_stex_current_module_str ? \l_stex_symdecl_name_str^^J
1860   Type:~\exp_not:o { \l_stex_symdecl_type_tl }^^J
1861   Args:~\prop_item:Nn \l_tmpa_prop { args }
1862 }
1863
1864 % circular dependencies require this:
1865
1866 \prop_if_exist:cF {
1867   l_stex_symdecl_
1868   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1869   _prop
1870 } {
1871   \prop_set_eq:cN {
1872     l_stex_symdecl_
1873     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1874     _prop
1875   } \l_tmpa_prop
1876 }
1877
1878 \seq_clear:c {
1879   l_stex_symdecl_
1880   \l_stex_current_module_str ? \l_stex_symdecl_name_str
1881   _notations

```



```

1882 }
1883
1884 \bool_if:NF \l_stex_symdecl_local_bool {
1885   \exp_args:Nx
1886   \stex_add_to_current_module:n {
1887     \seq_clear:c {
1888       l_stex_symdecl_
1889       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1890       _notations
1891     }
1892     \prop_set_from_keyval:cn {
1893       l_stex_symdecl_
1894       \l_stex_current_module_str ? \l_stex_symdecl_name_str
1895       _prop
1896     } {
1897       name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1898       module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1899       local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1900       type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1901       args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1902       arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1903       assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1904     }
1905   }
1906 }
1907
1908 \stex_if_smsmode:TF {
1909   \bool_if:NF \l_stex_symdecl_local_bool {
1910     % \exp_args:Nx \stex_add_to_sms:n {
1911     %   \prop_set_from_keyval:cn {
1912     %     l_stex_symdecl_
1913     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1914     %     _prop
1915     %   } {
1916     %     name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1917     %     module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1918     %     local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1919     %     type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1920     %     args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1921     %     arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1922     %     assocs    = \prop_item:Nn \l_tmpa_prop { assocs }    ,
1923     %   }
1924     %   \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1925     %     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1926     %   }
1927     % }
1928   }
1929 }{
1930   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1931     \l_stex_current_module_str ? \l_stex_symdecl_name_str
1932   }
1933   \stex_if_do_html:T {
1934     \stex_annotate_invisible:nnn {symdecl} {
1935       \l_stex_current_module_str ? \l_stex_symdecl_name_str

```

```

1936     } {
1937       \tl_if_empty:NF \l_stex_symdecl_type_tl {\stex_annotate_invisible:nnn{type}{}}{\l_st
1938       \stex_annotate_invisible:nnn{args}{}}{
1939         \prop_item:Nn \l_tmpa_prop { args }
1940       }
1941       \stex_annotate_invisible:nnn{macroname}{}{#1}
1942       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1943         \stex_annotate_invisible:nnn{definiens}{}
1944         {\l_stex_symdecl_definiens_tl$}
1945       }
1946     }
1947   }
1948 }
1949 }

```

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

`\stex_get_symbol:n`

```

1950 \str_new:N \l_stex_get_symbol_uri_str
1951
1952 \cs_new_protected:Nn \stex_get_symbol:n {
1953   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1954     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1955   }{
1956     % argument is a string
1957     % is it a command name?
1958     \cs_if_exist:cTF { #1 }{
1959       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1960       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1961       \str_if_empty:NTF \l_tmpa_str {
1962         \exp_args:Nx \cs_if_eq:NNTF {
1963           \tl_head:N \l_tmpa_tl
1964         } \stex_invoke_symbol:n {
1965           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1966         }{
1967           \__stex_symdecl_get_symbol_from_string:n { #1 }
1968         }
1969       } {
1970         \__stex_symdecl_get_symbol_from_string:n { #1 }
1971       }
1972     }{
1973       % argument is not a command name
1974       \__stex_symdecl_get_symbol_from_string:n { #1 }
1975       % \l_stex_all_symbols_seq
1976     }
1977   }
1978 }
1979
1980 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1981   \str_set:Nn \l_tmpa_str { #1 }
1982   \bool_set_false:N \l_tmpa_bool
1983   \stex_if_in_module:T {
1984     \exp_args:Nno \seq_if_in:cnT {c_stex_module\l_stex_current_module_str _constants} { \l
1985     \bool_set_true:N \l_tmpa_bool

```

```

1986     \str_set:Nx \l_stex_get_symbol_uri_str {
1987         \l_stex_current_module_str ? #1
1988     }
1989 }
1990 }
1991 \bool_if:NF \l_tmpa_bool {
1992     \tl_set:Nn \l_tmpa_tl {
1993         \msg_set:nnn{stex}{error/unknownsymbol}{
1994             No~symbol~#1~found!
1995         }
1996         \msg_error:nn{stex}{error/unknownsymbol}
1997     }
1998     \str_set:Nn \l_tmpa_str { #1 }
1999     \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
2000     \seq_map_inline:Nn \l_stex_all_symbols_seq {
2001         \str_set:Nn \l_tmpb_str { ##1 }
2002         \str_if_eq:eeT { \l_tmpa_str } {
2003             \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
2004         } {
2005             \seq_map_break:n {
2006                 \tl_set:Nn \l_tmpa_tl {
2007                     \str_set:Nn \l_stex_get_symbol_uri_str {
2008                         ##1
2009                     }
2010                 }
2011             }
2012         }
2013     }
2014     \l_tmpa_tl
2015 }
2016 }
2017
2018 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
2019     \exp_args:NNx \tl_set:Nn \l_tmpa_tl
2020     { \tl_tail:N \l_tmpa_tl }
2021     \tl_if_single:NTF \l_tmpa_tl {
2022         \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
2023             \exp_after:wN \str_set:Nn \exp_after:wN
2024             \l_stex_get_symbol_uri_str \l_tmpa_tl
2025         }{
2026             % TODO
2027             % tail is not a single group
2028         }
2029     }{
2030         % TODO
2031         % tail is not a single group
2032     }
2033 }

```

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

23.2 Notations

```

2034 <@@=stex_notation>

```

```

notation arguments:
2035 \keys_define:nn { stex / notation } {
2036   lang .tl_set_x:N = \l__stex_notation_lang_str ,
2037   variant .tl_set_x:N = \l__stex_notation_variant_str ,
2038   prec .str_set_x:N = \l__stex_notation_prec_str ,
2039   op .tl_set:N = \l__stex_notation_op_tl ,
2040   primary .bool_set:N = \l__stex_notation_primary_bool ,
2041   primary .default:n = {true} ,
2042   unknown .code:n = \str_set:Nx
2043     \l__stex_notation_variant_str \l_keys_key_str
2044 }
2045
2046 \cs_new_protected:Nn \__stex_notation_args:n {
2047   \str_clear:N \l__stex_notation_lang_str
2048   \str_clear:N \l__stex_notation_variant_str
2049   \str_clear:N \l__stex_notation_prec_str
2050   \tl_clear:N \l__stex_notation_op_tl
2051   \bool_set_false:N \l__stex_notation_primary_bool
2052
2053   \keys_set:nn { stex / notation } { #1 }
2054 }

```

\notation

```

2055 \NewDocumentCommand \notation { 0{ } m } {
2056   \__stex_notation_args:n { #1 }
2057   \tl_clear:N \l_stex_symdecl_definiens_tl
2058   \stex_get_symbol:n { #2 }
2059   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
2060 }
2061 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

2062 \cs_new_protected:Nn \stex_notation_do:nn {
2063   \let\l_stex_current_symbol_str\relax
2064   \prop_set_eq:Nc \l_tmpa_prop {
2065     l_stex_symdecl_ #1 _prop
2066   }
2067
2068   \prop_clear:N \l_tmpb_prop
2069   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
2070   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
2071   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
2072
2073   % precedences
2074   \seq_clear:N \l_tmpb_seq
2075   \exp_args:NNno
2076   \str_if_empty:NTF \l__stex_notation_prec_str {
2077     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2078     \int_compare:nNnTF \l_tmpa_str = 0 {
2079       \exp_args:NNnx
2080       \prop_put:Nno \l_tmpb_prop { opprec }
2081       { \neginfprec }
2082     }{

```

```

2083     \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2084   }
2085 } {
2086   \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
2087     \exp_args:NNnx
2088     \prop_put:Nno \l_tmpb_prop { opprec }
2089     { \neginfprec }
2090     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2091     \int_step_inline:nn { \l_tmpa_str } {
2092       \exp_args:NNx
2093       \seq_put_right:Nn \l_tmpb_seq { \infprec }
2094     }
2095   }{
2096     \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
2097     \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
2098       \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
2099       \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {
2100         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
2101         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
2102         \seq_map_inline:Nn \l_tmpa_seq {
2103           \seq_put_right:Nn \l_tmpb_seq { ##1 }
2104         }
2105       }
2106       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2107     }{
2108       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
2109       \int_compare:nNnTF \l_tmpa_str = 0 {
2110         \exp_args:NNnx
2111         \prop_put:Nno \l_tmpb_prop { opprec }
2112         { \infprec }
2113       }{
2114         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
2115       }
2116     }
2117   }
2118 }
2119
2120 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
2121 \int_step_inline:nn { \l_tmpa_str } {
2122   \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
2123     \exp_args:NNx
2124     \seq_put_right:Nn \l_tmpb_seq {
2125       \prop_item:Nn \l_tmpb_prop { opprec }
2126     }
2127   }
2128 }
2129
2130 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
2131 \tl_clear:N \l_tmpa_tl
2132
2133 \int_compare:nNnTF \l_tmpa_str = 0 {
2134   \exp_args:NNe
2135   \cs_set:Npn \l__stex_notation_macrocode_cs {
2136     \_stex_term_math_oms:nnnn { \l_stex_current_symbol_str }

```

```

2137     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2138     { \prop_item:Nn \l_tmpb_prop { opprec } }
2139     { \exp_not:n { #2 } }
2140   }
2141   \__stex_notation_final:
2142 }{
2143   \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
2144   \str_if_in:NnTF \l_tmpb_str b {
2145     \exp_args:Nne \use:nn
2146     {
2147       \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2148       \cs_set:Npn \l_tmpa_str } { {
2149         \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2150         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2151         { \prop_item:Nn \l_tmpb_prop { opprec } }
2152         { \exp_not:n { #2 } }
2153       } }
2154     }{
2155       \str_if_in:NnTF \l_tmpb_str B {
2156         \exp_args:Nne \use:nn
2157         {
2158           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2159           \cs_set:Npn \l_tmpa_str } { {
2160             \stex_term_math_omb:nnnn { \l_stex_current_symbol_str }
2161             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2162             { \prop_item:Nn \l_tmpb_prop { opprec } }
2163             { \exp_not:n { #2 } }
2164           } }
2165         }{
2166           \exp_args:Nne \use:nn
2167           {
2168             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2169             \cs_set:Npn \l_tmpa_str } { {
2170               \stex_term_math_oma:nnnn { \l_stex_current_symbol_str }
2171               { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2172               { \prop_item:Nn \l_tmpb_prop { opprec } }
2173               { \exp_not:n { #2 } }
2174             } }
2175           }
2176         }
2177       \int_zero:N \l_tmpa_int
2178       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2179       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2180       \__stex_notation_arguments:
2181     }
2182   }
2183 }

```

(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

```

2184 \cs_new_protected:Nn \__stex_notation_arguments: {
2185   \int_incr:N \l_tmpa_int
2186   \str_if_empty:NTF \l_tmpa_str {

```

```

2187   \_stex_notation_final:
2188   }{
2189     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2190     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2191     \str_if_eq:VnTF \l_tmpb_str a {
2192       \_stex_notation_argument_assoc:n
2193     }{
2194       \str_if_eq:VnTF \l_tmpb_str B {
2195         \_stex_notation_argument_assoc:n
2196       }{
2197         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2198         \tl_put_right:Nx \l_tmpa_tl {
2199           { \_stex_term_math_arg:nnn
2200             { \int_use:N \l_tmpa_int }
2201             { \l_tmpb_str }
2202             { ####\int_use:N \l_tmpa_int }
2203           }
2204         }
2205         \_stex_notation_arguments:
2206       }
2207     }
2208   }
2209 }

```

(End definition for _stex_notation_arguments:.)

_stex_notation_argument_assoc:n

```

2210 \cs_new_protected:Nn \_stex_notation_argument_assoc:n {
2211   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2212   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2213   \tl_put_right:Nx \l_tmpa_tl {
2214     { \_stex_term_math_assoc_arg:nnnn
2215       { \int_use:N \l_tmpa_int }
2216       { \l_tmpb_str }
2217       \exp_args:No \exp_not:n
2218       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2219       { ####\int_use:N \l_tmpa_int }
2220     }
2221   }
2222   \_stex_notation_arguments:
2223 }

```

(End definition for _stex_notation_argument_assoc:n.)

_stex_notation_final: Called after processing all notation arguments

```

2224 \cs_new_protected:Nn \_stex_notation_final: {
2225   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2226   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2227   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2228   \exp_args:Nne \use:nn
2229   {
2230     \cs_generate_from_arg_count:cNnn {
2231       stex_notation_ \l_tmpa_str \c_hash_str
2232       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str

```

```

2233     _cs
2234 }
2235 \cs_set:Npn \l_tmpb_str } { {
2236     \exp_after:wN \exp_after:wN \exp_after:wN
2237     \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2238     { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2239 } }
2240
2241 \tl_if_empty:NF \l__stex_notation_op_tl {
2242     \cs_set:cpx {
2243         stex_op_notation_ \l_tmpa_str \c_hash_str
2244         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2245         _cs
2246     } {
2247         \stex_term_oms:nnn {
2248             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2249             \l__stex_notation_lang_str
2250         }{
2251             \l_tmpa_str
2252         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2253     }
2254 }
2255
2256 \exp_args:Ne
2257 \stex_add_to_current_module:n {
2258     \cs_generate_from_arg_count:cNnn {
2259         stex_notation_ \l_tmpa_str \c_hash_str
2260         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2261         _cs
2262     } \cs_set:Npn {\l_tmpb_str} { {
2263         \exp_after:wN \exp_after:wN \exp_after:wN
2264         \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2265         { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2266     } }
2267 \tl_if_empty:NF \l__stex_notation_op_tl {
2268     \cs_set:cpn {
2269         stex_op_notation_ \l_tmpa_str \c_hash_str
2270         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2271         _cs
2272     } {
2273         \stex_term_oms:nnn {
2274             \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2275             \l__stex_notation_lang_str
2276         }{
2277             \l_tmpa_str
2278         }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2279     }
2280 }
2281 }
2282
2283 \seq_put_right:cx {
2284     l_stex_symdecl_
2285     \prop_item:Nn \l_tmpb_prop { symbol }
2286     _notations

```



```

2287 } {
2288   \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2289 }
2290
2291 \stex_debug:nn{symbols}{
2292   Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2293   ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2294   Operator~precedence:~
2295   \prop_item:Nn \l_tmpb_prop { opprec }^^J
2296   Argument~precedences:~
2297   \seq_use:Nn \l_tmpa_seq {,~}^^J
2298   Notation: \cs_meaning:c {
2299     stex_notation_ \l_tmpa_str \c_hash_str
2300     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2301     _cs
2302   }
2303 }
2304
2305 \prop_set_eq:cN {
2306   l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2307   \c_hash_str \l__stex_notation_lang_str _prop
2308 } \l_tmpb_prop
2309
2310 \exp_args:Ne
2311 \stex_add_to_current_module:n {
2312   \seq_put_right:cn {
2313     l_stex_symdecl_
2314     \prop_item:Nn \l_tmpb_prop { symbol }
2315     _notations
2316   } {
2317     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2318   }
2319   \prop_set_from_keyval:cn {
2320     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2321     \c_hash_str \l__stex_notation_lang_str _prop
2322   } {
2323     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2324     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2325     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,
2326     opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2327     argprecs    = \prop_item:Nn \l_tmpb_prop { argprecs }    ,
2328   }
2329 }
2330
2331 \stex_if_smsmode:TF {
2332   \stex_smsmode_set_codes:
2333   % \exp_args:Nx \stex_add_to_sms:n {
2334   %   \prop_set_from_keyval:cn {
2335   %     l_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2336   %     \c_hash_str \l__stex_notation_lang_str _prop
2337   %   } {
2338   %     symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2339   %     language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2340   %     variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2341 %      opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2342 %      argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2343 %    }
2344 %  }
2345 }{
2346
2347 % HTML annotations
2348 \stex_if_do_html:T {
2349   \stex_annotate_invisible:nnn { notation }
2350   { \prop_item:Nn \l_tmpb_prop { symbol } } {
2351     \stex_annotate_invisible:nnn { notationfragment }
2352     { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2353     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2354     \stex_annotate_invisible:nnn { precedence }
2355     { \prop_item:Nn \l_tmpb_prop { opprec };
2356       \seq_use:Nn \l_tmpa_seq { x }
2357     }{}
2358
2359     \int_zero:N \l_tmpa_int
2360     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2361     \tl_clear:N \l_tmpa_tl
2362     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2363       \int_incr:N \l_tmpa_int
2364       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2365       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2366       \str_if_eq:VnTF \l_tmpb_str a {
2367         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2368           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2369           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2370         } }
2371       }{
2372         \str_if_eq:VnTF \l_tmpb_str B {
2373           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2374             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2375             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2376           } }
2377         }{
2378           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2379             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2380           } }
2381         }
2382       }
2383     }
2384     \stex_annotate_invisible:nnn { notationcomp }{}{
2385       $ \exp_args:Nno \use:nn { \use:c {
2386         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }
2387         \c_hash_str \l__stex_notation_variant_str
2388         \c_hash_str \l__stex_notation_lang_str _cs
2389       } } { \l_tmpa_tl } $
2390     }
2391   }
2392 }
2393 }
2394 }

```

(End definition for `_stex_notation_final:`)

`\symdef`

```

2395 \keys_define:nn { stex / symdef } {
2396   name      .str_set_x:N = \l_stex_symdecl_name_str ,
2397   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2398   args      .str_set_x:N = \l_stex_symdecl_args_str ,
2399   type      .tl_set:N = \l_stex_symdecl_type_tl ,
2400   def       .tl_set:N = \l_stex_symdecl_definiens_tl ,
2401   op        .tl_set:N = \l__stex_notation_op_tl ,
2402   lang      .str_set_x:N = \l__stex_notation_lang_str ,
2403   variant   .str_set_x:N = \l__stex_notation_variant_str ,
2404   prec      .str_set_x:N = \l__stex_notation_prec_str ,
2405   unknown   .code:n = \str_set:Nx
2406             \l__stex_notation_variant_str \l_keys_key_str
2407 }
2408
2409 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2410   \str_clear:N \l_stex_symdecl_name_str
2411   \str_clear:N \l_stex_symdecl_args_str
2412   \bool_set_false:N \l_stex_symdecl_local_bool
2413   \tl_clear:N \l_stex_symdecl_type_tl
2414   \tl_clear:N \l_stex_symdecl_definiens_tl
2415   \str_clear:N \l__stex_notation_lang_str
2416   \str_clear:N \l__stex_notation_variant_str
2417   \str_clear:N \l__stex_notation_prec_str
2418   \tl_clear:N \l__stex_notation_op_tl
2419
2420   \keys_set:nn { stex / symdef } { #1 }
2421 }
2422
2423 \NewDocumentCommand \symdef { 0{} m } {
2424   \_stex_notation_symdef_args:n { #1 }
2425   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2426   \stex_symdecl_do:n { #2 }
2427   \exp_args:Nx \stex_notation_do:nn {
2428     \l_stex_current_module_str ? \l_stex_symdecl_name_str
2429   }
2430 }
2431 \stex_deactivate_macro:Nn \symdef {module~environments}
2432
2433 (End definition for \symdef. This function is documented on page 26.)
2434 \endpackage

```

Chapter 24

STEX -Terms Implementation

```
2433 <*package>
2434
2435 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2436
2437 <@@=stex_terms>
2438
2439 Warnings and error messages
2440 \msg_new:nnn{stex}{error/nonotation}{
2441   Symbol~#1~invoked,~but~has~no~notation#2!
2442 }
2443 \msg_new:nnn{stex}{error/notationarg}{
2444   Error~in~parsing~notation~#1
2445 }
2446 \msg_new:nnn{stex}{error/noop}{
2447   Symbol~#1~has~no~operator~notation~for~notation~#2
2448 }
```

24.1 Symbol Invocations

Arguments:

```
2448 \keys_define:nn { stex / terms } {
2449   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2450   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2451   unknown .code:n = \str_set:Nx
2452     \l__stex_terms_variant_str \l_keys_key_str
2453 }
2454
2455 \cs_new_protected:Nn \__stex_terms_args:n {
2456   \str_clear:N \l__stex_terms_lang_str
2457   \str_clear:N \l__stex_terms_variant_str
2458   \str_clear:N \l__stex_terms_prec_str
2459   \tl_clear:N \l__stex_terms_op_tl
2460
2461   \keys_set:nn { stex / terms } { #1 }
```

2462 }

\stex_invoke_symbol:n Invokes a semantic macro

```
2463 \cs_new_protected:Nn \stex_invoke_symbol:n {
2464   \if_mode_math:
2465     \exp_after:wN \__stex_terms_invoke_math:n
2466   \else:
2467     \exp_after:wN \__stex_terms_invoke_text:n
2468   \fi: { #1 }
2469 }
```

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

__stex_terms_invoke_math:n

```
2470 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2471   \peek_charcode_remove:NTF ! {
2472     \peek_charcode:NTF [ {
2473       \__stex_terms_invoke_op:nw { #1 }
2474     }{
2475       \peek_charcode_remove:NTF ! {
2476         \peek_charcode:NTF [ {
2477           \__stex_terms_invoke_op_custom:nw
2478         }{
2479           % TODO throw error
2480         }
2481       }{
2482         \__stex_terms_invoke_op:nw { #1 } []
2483       }
2484     }{
2485       \peek_charcode_remove:NTF * {
2486         \__stex_terms_invoke_text:n { #1 }
2487       }{
2488         \peek_charcode:NTF [ {
2489           \__stex_terms_invoke_math:nw { #1 }
2490         }{
2491           \__stex_terms_invoke_math:nw { #1 } []
2492         }
2493       }
2494     }
2495   }
2496 }
```

(End definition for __stex_terms_invoke_math:n.)

__stex_terms_invoke_op_custom:nw

```
2497 \cs_new_protected:Npn \__stex_terms_invoke_op_custom:nw #1 [#2] {
2498   \stex_term_oms:nnn {#1 \c_hash_str\c_hash_str}{#1}{
2499     \stex_highlight_term:nn{#1}{#2}
2500   }
2501 }
```

(End definition for __stex_terms_invoke_op_custom:nw.)

_stex_terms_invoke_op:nw

```

2502 \cs_new_protected:Npn \_stex_terms_invoke_op:nw #1 [#2] {
2503   \_stex_terms_args:n { #2 }
2504   \cs_if_exist:cTF {
2505     stex_op_notation_ #1 \c_hash_str
2506     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2507   }{
2508     \csname stex_op_notation_ #1 \c_hash_str
2509       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str _cs
2510     \endcsname
2511   }{
2512     \msg_error:nnxx{stex}{error/noop}{#1}{\l__stex_terms_variant_str \c_hash_str \l__stex_te
2513   }
2514 }

```

(End definition for _stex_terms_invoke_op:nw.)

_stex_terms_invoke_math:nw

```

2515 \cs_new_protected:Npn \_stex_terms_invoke_math:nw #1 [#2] {
2516   \_stex_terms_args:n { #2 }
2517   \seq_if_empty:cTF {
2518     l_stex_symdecl_ #1 _notations
2519   } {
2520     \msg_error:nnxx{stex}{error/nonotation}{#1}{s}
2521   } {
2522     \seq_if_in:cxTF {
2523       l_stex_symdecl_ #1 _notations
2524     }
2525     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2526       \str_set:Nn \l_stex_current_symbol_str { #1 }
2527       \use:c{
2528         stex_notation_ #1 \c_hash_str
2529         \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2530         _cs
2531       }
2532     }{
2533       \str_if_empty:NTF \l__stex_terms_variant_str {
2534         \str_if_empty:NTF \l__stex_terms_lang_str {
2535           \seq_get_left:cN {
2536             l_stex_symdecl_ #1 _notations
2537           } \l_tmpa_str
2538           \str_set:Nn \l_stex_current_symbol_str { #1 }
2539           \use:c{
2540             stex_notation_ #1 \c_hash_str \l_tmpa_str
2541             _cs
2542           }
2543         }{
2544           \msg_error:nnxx{stex}{error/nonotation}{#1}{
2545             ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2546           }
2547         }
2548       }{
2549         \msg_error:nnxx{stex}{error/nonotation}{#1}{
2550           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str

```

```

2551     }
2552   }
2553 }
2554 }
2555 }

```

(End definition for `_stex_terms_invoke_math:nw`.)

`_stex_terms_invoke_text:n`

```

2556 \cs_new_protected:Nn \_stex_terms_invoke_text:n {
2557   \peek_charcode_remove:NTF ! {
2558     \stex_term_custom:nn { #1 } { }
2559   }{
2560     \prop_set_eq:Nc \l_tmpa_prop {
2561       l_stex_symdecl_ #1 _prop
2562     }
2563     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2564     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2565   }
2566 }

```

(End definition for `_stex_terms_invoke_text:n`.)

24.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2567 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2568 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2569 \int_new:N \l__stex_terms_downprec
2570 \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
2571 \tl_set:Nn \l__stex_terms_left_bracket_str (
2572 \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

```

2573 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2574   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2575     \bool_set_false:N \l__stex_terms_brackets_done_bool
2576     #2
2577   } {
2578     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2579       \bool_if:NTF \l_stex_inarray_bool { #2 }{
2580         \stex_debug:nn{dobrackets}{\number#1 > \number\l__stex_terms_downprec; \detokenize{#
2581         \dobrackets { #2 }
2582       }

```

```

2583     }{ #2 }
2584   }
2585 }

```

(End definition for `_stex_terms_maybe_brackets:nn`.)

`\dobrackets`

```

2586 \bool_new:N \l__stex_terms_brackets_done_bool
2587 %\RequirePackage{scalerel}
2588 \cs_new_protected:Npn \dobrackets #1 {
2589   %\ThisStyle{\if D\m@switch
2590   %   \exp_args:Nnx \use:nn
2591   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2592   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2593   % \else
2594   \exp_args:Nnx \use:nn
2595   {
2596     \bool_set_true:N \l__stex_terms_brackets_done_bool
2597     \int_set:Nn \l__stex_terms_downprec \infprec
2598     \l__stex_terms_left_bracket_str
2599     #1
2600   }
2601   {
2602     \bool_set_false:N \l__stex_terms_brackets_done_bool
2603     \l__stex_terms_right_bracket_str
2604     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2605   }
2606   %\fi}
2607 }

```

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

`\withbrackets`

```

2608 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2609   \exp_args:Nnx \use:nn
2610   {
2611     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2612     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2613     #3
2614   }
2615   {
2616     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2617     {\l__stex_terms_left_bracket_str}
2618     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2619     {\l__stex_terms_right_bracket_str}
2620   }
2621 }

```

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

`\STEXinvisible`

```

2622 \cs_new_protected:Npn \STEXinvisible #1 {
2623   \stex_annotate_invisible:n { #1 }
2624 }

```


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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2625 \cs_new_protected:Nn \_stex_term_oms:nnn {
2626   \stex_annotate:nnn{ OMID }{ #2 }{
2627     \stex_highlight_term:nn { #1 } { #3 }
2628   }
2629 }
2630
2631 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2632   \__stex_terms_maybe_brackets:nn { #3 }{
2633     \stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2634   }
2635 }

```

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

`_stex_term_math_oma:nnnn`

```

2636 \cs_new_protected:Nn \_stex_term_oma:nnn {
2637   \stex_annotate:nnn{ OMA }{ #2 }{
2638     \stex_highlight_term:nn { #1 } { #3 }
2639   }
2640 }
2641
2642 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2643   \__stex_terms_maybe_brackets:nn { #3 }{
2644     \stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2645   }
2646 }

```

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

`_stex_term_math_omb:nnnn`

```

2647 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2648   \stex_annotate:nnn{ OMBIND }{ #2 }{
2649     \stex_highlight_term:nn { #1 } { #3 }
2650   }
2651 }
2652
2653 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2654   \__stex_terms_maybe_brackets:nn { #3 }{
2655     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2656   }
2657 }

```

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

`_stex_term_math_arg:nnn`

```

2658 \cs_new_protected:Nn \_stex_term_arg:nn {
2659   \stex_unhighlight_term:n {
2660     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2661   }
2662 }

```

```

2663 \cs_new_protected:Nn \stex_term_math_arg:nnn {
2664   \exp_args:Nnx \use:nn
2665     { \int_set:Nn \l__stex_terms_downprec { #2 }
2666       \stex_term_arg:nn { #1 }{ #3 }
2667     }
2668     { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2669   }

```

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

`\stex_term_math_assoc_arg:nnnn`

```

2670 \cs_new_protected:Nn \stex_term_math_assoc_arg:nnnn {
2671   \clist_set:Nn \l_tmpa_clist{ #4 }
2672   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2673     \tl_set:Nn \l_tmpa_tl { #4 }
2674   }{
2675     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2676     \clist_reverse:N \l_tmpa_clist
2677     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2678
2679     \clist_map_inline:Nn \l_tmpa_clist {
2680       \exp_args:NNNo \exp_args:Nno \tl_set:No \l_tmpa_tl {
2681         \exp_args:Nno
2682           \l_tmpa_cs { ##1 } \l_tmpa_tl
2683       }
2684     }
2685
2686   }
2687   \exp_args:Nnno
2688   \stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2689 }

```

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

`\stex_term_custom:nn`

```

2690 \cs_new_protected:Nn \stex_term_custom:nn {
2691   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2692   \str_set:Nn \l_tmpa_str { #2 }
2693   \tl_clear:N \l_tmpa_tl
2694   \int_zero:N \l_tmpa_int
2695   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2696   \__stex_terms_custom_loop:
2697 }

```

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

`__stex_terms_custom_loop:`

```

2698 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2699   \bool_set_false:N \l_tmpa_bool
2700   \bool_while_do:nn {
2701     \str_if_eq_p:ee X {
2702       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2703     }
2704   }{
2705     \int_incr:N \l_tmpa_int

```

```

2706 }
2707
2708 \peek_charcode:NTF [ {
2709   % notation/text component
2710   \__stex_terms_custom_component:w
2711 } {
2712   \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2713     % all arguments read => finish
2714     \__stex_terms_custom_final:
2715   } {
2716     % arguments missing
2717     \peek_charcode_remove:NTF * {
2718       % invisible, specific argument position or both
2719       \peek_charcode:NTF [ {
2720         % visible specific argument position
2721         \__stex_terms_custom_arg:wn
2722       } {
2723         % invisible
2724         \peek_charcode_remove:NTF * {
2725           % invisible specific argument position
2726           \__stex_terms_custom_arg_inv:wn
2727         } {
2728           % invisible next argument
2729           \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2730         }
2731       }
2732     } {
2733       % next normal argument
2734       \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2735     }
2736   }
2737 }
2738 }

```

(End definition for __stex_terms_custom_loop:.)

__stex_terms_custom_arg_inv:wn

```

2739 \cs_new_protected:Npn \__stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2740   \bool_set_true:N \l_tmpa_bool
2741   \__stex_terms_custom_arg:wn [ #1 ] { #2 }
2742 }

```

(End definition for __stex_terms_custom_arg_inv:wn.)

__stex_terms_custom_arg:wn

```

2743 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2744   \str_set:Nx \l_tmpb_str {
2745     \str_item:Nn \l_tmpa_str { #1 }
2746   }
2747   \str_case:VnTF \l_tmpb_str {
2748     { X } {
2749       \msg_error:nnx{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2750     }
2751     { i } { \__stex_terms_custom_set_X:n { #1 } }
2752     { b } { \__stex_terms_custom_set_X:n { #1 } }

```

```

2753 { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2754 { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2755 }{}{
2756 \msg_error:nnx{stex}{error/notationarg}{\l\_stex_terms_custom_uri}
2757 }
2758
2759 \bool_if:nTF \l_tmpa_bool {
2760 \tl_put_right:Nx \l_tmpa_tl {
2761 \stex_annotate_invisible:n {
2762 \stex_term_arg:nn { \int_eval:n { #1 } }
2763 \exp_not:n { { #2 } }
2764 }
2765 }
2766 } {
2767 \tl_put_right:Nx \l_tmpa_tl {
2768 \stex_term_arg:nn { \int_eval:n { #1 } }
2769 \exp_not:n { { #2 } }
2770 }
2771 }
2772
2773 \_stex_terms_custom_loop:
2774 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2775 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2776 \str_set:Nx \l_tmpa_str {
2777 \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2778 X
2779 \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2780 }
2781 }

```

(End definition for _stex_terms_custom_set_X:n.)

_stex_terms_custom_component:

```

2782 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2783 \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2784 \_stex_terms_custom_loop:
2785 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2786 \cs_new_protected:Nn \_stex_terms_custom_final: {
2787 \int_compare:nNnTF \l_tmpb_int = 0 {
2788 \exp_args:Nnno \stex_term_oms:nnn
2789 }{
2790 \str_if_in:NnTF \l_tmpa_str {b} {
2791 \exp_args:Nnno \stex_term_ombind:nnn
2792 } {
2793 \exp_args:Nnno \stex_term_oma:nnn
2794 }
2795 }

```

```

2796 { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2797 }

```

(End definition for `_stex_terms_custom_final:`.)

\symref

\symname

```

2798 \NewDocumentCommand \symref { m m }{
2799   \let\compemph_uri_prev:\compemph@uri
2800   \let\compemph@uri\symrefemph@uri
2801   \STEXsymbol{#1}! [#2]
2802   \let\compemph@uri\compemph_uri_prev:
2803 }
2804
2805 \keys_define:nn { stex / symname } {
2806   post      .str_set_x:N    = \l_stex_symname_post_str
2807 }
2808
2809 \cs_new_protected:Nn \stex_symname_args:n {
2810   \str_clear:N \l_stex_symname_post_str
2811   \keys_set:nn { stex / symname } { #1 }
2812 }
2813
2814 \NewDocumentCommand \symname { 0{} m }{
2815   \stex_symname_args:n { #1 }
2816   \stex_get_symbol:n { #2 }
2817   \str_set:Nx \l_tmpa_str {
2818     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2819   }
2820   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2821
2822   \let\compemph_uri_prev:\compemph@uri
2823   \let\compemph@uri\symrefemph@uri
2824   \exp_args:NNx \use:nn
2825   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }! [
2826     \l_tmpa_str \l_stex_symname_post_str
2827   ] }
2828   \let\compemph@uri\compemph_uri_prev:
2829 }

```

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

24.3 Notation Components

```

2830 <@@=stex_notationcomps>

```

\stex_highlight_term:nn

```

2831
2832 \str_new:N \l_stex_current_symbol_str
2833 \cs_new_protected:Nn \stex_highlight_term:nn {
2834   \exp_args:Nnx
2835   \use:nn {
2836     \str_set:Nx \l_stex_current_symbol_str { #1 }
2837     #2
2838   } {

```

```

2839 \str_set:Nx \exp_not:N \l_stex_current_symbol_str
2840 { \l_stex_current_symbol_str }
2841 }
2842 }
2843
2844 \cs_new_protected:Nn \stex_unhighlight_term:n {
2845 % \latexml_if:TF {
2846 % #1
2847 % } {
2848 % \rustex_if:TF {
2849 % #1
2850 % } {
2851 % #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2852 % }
2853 % }
2854 }

```

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

```

\comp
\compemph@uri 2855 \cs_new_protected:Npn \comp #1 {
\compemph 2856 \str_if_empty:NF \l_stex_current_symbol_str {
\defemph 2857 \rustex_if:TF {
\defemph@uri 2858 \stex_annotate:nnn { comp }{ \l_stex_current_symbol_str }{ #1 }
\symrefemph 2859 }{
\symrefemph@uri 2860 \exp_args:Nnx \compemph@uri { #1 } { \l_stex_current_symbol_str }
2861 }
2862 }
2863 }
2864
2865 \cs_new_protected:Npn \compemph@uri #1 #2 {
2866 \compemph{ #1 }
2867 }
2868
2869
2870 \cs_new_protected:Npn \compemph #1 {
2871 \textcolor{blue}{#1}
2872 }
2873
2874 \cs_new_protected:Npn \defemph@uri #1 #2 {
2875 \defemph{#1}
2876 }
2877
2878 \cs_new_protected:Npn \defemph #1 {
2879 \textbf{#1}
2880 }
2881
2882 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2883 \symrefemph{#1}
2884 }
2885
2886 \cs_new_protected:Npn \symrefemph #1 {
2887 \textbf{#1}
2888 }

```

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

`\ellipses`

```
2889 \NewDocumentCommand \ellipses {} { \ldots }
```

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

```

\parray
\prmatrix 2890 \bool_new:N \l_stex_inarray_bool
\parrayline 2891 \bool_set_false:N \l_stex_inarray_bool
\parraylineh 2892 \NewDocumentCommand \parray { m m } {
\parraycell 2893 \begingroup
2894 \bool_set_true:N \l_stex_inarray_bool
2895 \begin{array}{#1}
2896 #2
2897 \end{array}
2898 \endgroup
2899 }
2900
2901 \NewDocumentCommand \prmatrix { m } {
2902 \begingroup
2903 \bool_set_true:N \l_stex_inarray_bool
2904 \begin{matrix}
2905 #1
2906 \end{matrix}
2907 \endgroup
2908 }
2909
2910 \def \maybepline {
2911 \bool_if:NT \l_stex_inarray_bool {\hline}
2912 }
2913
2914 \def \parrayline #1 #2 {
2915 #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2916 }
2917
2918 \def \pmrow #1 { \parrayline{}{ #1 } }
2919
2920 \def \parraylineh #1 #2 {
2921 #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2922 }
2923
2924 \def \parraycell #1 {
2925 #1 \bool_if:NT \l_stex_inarray_bool {&}
2926 }

```

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

```
2927 \endpackage
```

Chapter 25

STEX -Structural Features Implementation

```
2928 <*package>
2929
2930 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2931
2932 <@@=stex_features>
2933
2934 Warnings and error messages
```

25.1 Imports with modification

```
2934 \seq_new:N \l_stex_implicit_morphisms_seq
2935 \NewDocumentCommand \implicitmorphism { 0{} m m }{
2936   \stex_import_module_uri:nn { #1 } { #2 }
2937   \stex_debug:nn{implicits}{
2938     Implicit~morphism:~
2939     \l_stex_module_ns_str ? \l__stex_features_name_str
2940   }
2941   \exp_args:NNx \seq_if_in:NnT \l_stex_all_modules_seq {
2942     \l_stex_module_ns_str ? \l__stex_features_name_str
2943   }{
2944     \msg_error:nnn{stex}{error/conflictingmodules}{
2945       \l_stex_module_ns_str ? \l__stex_features_name_str
2946     }
2947   }
2948
2949   % TODO
2950
2951
2952
2953 \seq_put_right:Nx \l_stex_implicit_morphisms_seq {
2954   \l_stex_module_ns_str ? \l__stex_features_name_str
2955 }
```



```

2956 }
2957

```

25.2 The feature environment

structural@feature

```

2958
2959 \NewDocumentEnvironment{structural@feature}{ m m m }{
2960   \stex_if_in_module:F {
2961     \msg_set:nnn{stex}{error/nomodule}{
2962       Structural-Feature-has-to-occur-in-a-module:\\
2963       Feature-#2-of-type-#1\\
2964       In-File:~\stex_path_to_string:N \g_stex_currentfile_seq
2965     }
2966     \msg_error:nn{stex}{error/nomodule}
2967   }
2968
2969   \str_set:Nx \l_stex_module_name_str {
2970     \prop_item:Nn \l_stex_current_module_prop
2971       { name } / #2 - feature
2972   }
2973
2974   \str_set:Nx \l_stex_module_ns_str {
2975     \prop_item:Nn \l_stex_current_module_prop
2976       { ns }
2977   }
2978
2979
2980   \str_clear:N \l_tmpa_str
2981   \seq_clear:N \l_tmpa_seq
2982   \tl_clear:N \l_tmpa_tl
2983   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2984     origname = #2,
2985     name      = \l_stex_module_name_str ,
2986     ns        = \l_stex_module_ns_str ,
2987     imports   = \exp_not:o { \l_tmpa_seq } ,
2988     constants = \exp_not:o { \l_tmpa_seq } ,
2989     content   = \exp_not:o { \l_tmpa_tl } ,
2990     file      = \exp_not:o { \g_stex_currentfile_seq } ,
2991     lang      = \l_stex_module_lang_str ,
2992     sig       = \l_tmpa_str ,
2993     meta      = \l_tmpa_str ,
2994     feature   = #1 ,
2995   }
2996
2997   \stex_if_smsmode:TF {
2998     \stex_smsmode_set_codes:
2999   } {
3000     \begin{stex_annotate_env}{ feature:#1 }{}
3001     \stex_annotate_invisible:nnn{header}{}{ #3 }
3002   }
3003 }{
3004   \str_set:Nx \l_tmpa_str {
3005     c_stex_feature_

```

```

3006     \prop_item:Nn \l_stex_current_module_prop { ns } ?
3007     \prop_item:Nn \l_stex_current_module_prop { name }
3008     _prop
3009 }
3010 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
3011 \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
3012 \stex_if_smsmode:TF {
3013     \exp_args:Nx \stex_add_to_sms:n {
3014         \prop_gset_from_keyval:cn {
3015             c_stex_feature_
3016             \prop_item:Nn \l_stex_current_module_prop { ns } ?
3017             \prop_item:Nn \l_stex_current_module_prop { name }
3018             _prop
3019         } {
3020             origname = #2,
3021             name     = \prop_item:cn { \l_tmpa_str } { name } ,
3022             ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
3023             imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
3024             constants = \prop_item:cn { \l_tmpa_str } { constants } ,
3025             content  = \prop_item:cn { \l_tmpa_str } { content } ,
3026             file     = \prop_item:cn { \l_tmpa_str } { file } ,
3027             lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
3028             sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
3029             meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
3030             feature  = \prop_item:cn { \l_tmpa_str } { feature }
3031         }
3032     }
3033 } {
3034     \end{stex_annotate_env}
3035 }
3036 }
3037

```

25.3 Features

structure

```

3038
3039 \prop_new:N \l_stex_all_structures_prop
3040
3041 \keys_define:nn { stex / features / structure } {
3042     name .str_set_x:N = \l__stex_features_structure_name_str ,
3043 }
3044
3045 \cs_new_protected:Nn \__stex_features_structure_args:n {
3046     \str_clear:N \l__stex_features_structure_name_str
3047     \keys_set:nn { stex / features / structure } { #1 }
3048 }
3049
3050 %\stex_new_feature:nnnn { structure } { 0{ } m } {
3051 % \__stex_features_structure_args:n { ##1 }
3052 % \str_if_empty:NT \l__stex_features_structure_name_str {
3053 % \str_set:Nx \l__stex_features_structure_name_str { ##2 }
3054 % }

```

```

3055 %} {
3056 %
3057 %}
3058
3059 \NewDocumentEnvironment{mathstructure}{0}{m}{
3060   \_stex_features_structure_args:n { #1 }
3061   \str_if_empty:NT \_l\_stex_features_structure_name_str {
3062     \str_set:Nx \_l\_stex_features_structure_name_str { #2 }
3063   }
3064   \exp_args:Nnnx
3065   \begin{structural@feature}{ structure }
3066     { \_l\_stex_features_structure_name_str }{}
3067     \seq_clear:N \_l\_tmpa_seq
3068     \prop_put:Nno \_l\_stex_current_module_prop { fields } \_l\_tmpa_seq
3069
3070   }{
3071     \prop_get:NnN \_l\_stex_current_module_prop { constants } \_l\_tmpa_seq
3072     \prop_get:NnN \_l\_stex_current_module_prop { fields } \_l\_tmpb_seq
3073     \str_set:Nx \_l\_tmpa_str {
3074       \prop_item:Nn \_l\_stex_current_module_prop { ns } ?
3075       \prop_item:Nn \_l\_stex_current_module_prop { name }
3076     }
3077     \seq_map_inline:Nn \_l\_tmpa_seq {
3078       \exp_args:NNx \seq_put_right:Nn \_l\_tmpb_seq { \_l\_tmpa_str ? ##1 }
3079     }
3080     \prop_put:Nno \_l\_stex_current_module_prop { fields } { \_l\_tmpb_seq }
3081     \exp_args:Nnx
3082     \AddToHookNext { env / mathstructure / after }{
3083       \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}}{
3084         \_stex_term_math_oms:nnnn { \_l\_tmpa_str }{}{0}{}
3085       }}, name = \prop_item:Nn \_l\_stex_current_module_prop { origname }]{ #2 }
3086       \STEXexport {
3087         \prop_put:Nno \exp_not:N \_l\_stex_all_structures_prop
3088           {\prop_item:Nn \_l\_stex_current_module_prop { origname }}
3089           {\_l\_tmpa_str}
3090         \prop_put:Nno \exp_not:N \_l\_stex_all_structures_prop
3091           {#2}{\_l\_tmpa_str}
3092         % \seq_put_right:Nn \exp_not:N \_l\_stex_all_structures_seq {
3093         %   \prop_item:Nn \_l\_stex_current_module_prop { origname },
3094         %   \_l\_tmpa_str
3095         % }
3096         % \seq_put_right:Nn \exp_not:N \_l\_stex_all_structures_seq {
3097         %   #2,\_l\_tmpa_str
3098         % }
3099         % \tl_set:cx { #2 } {
3100         %   \stex_invoke_structure:n { \_l\_tmpa_str }
3101         % }
3102       }
3103
3104   \end{structural@feature}
3105   % \g_stex_last_feature_prop
3106 }

```

\instantiate

```

3107 \seq_new:N \l__stex_features_structure_field_seq
3108 \str_new:N \l__stex_features_structure_field_str
3109 \str_new:N \l__stex_features_structure_def_tl
3110 \prop_new:N \l__stex_features_structure_prop
3111 \NewDocumentCommand \instantiate { m O{} m }{
3112   \stex_smsmode_set_codes:
3113   \prop_get:NnN \l__stex_all_structures_prop {#1} \l_tmpa_str
3114   \prop_set_eq:Nc \l__stex_features_structure_prop {
3115     c_stex_feature_\l_tmpa_str _prop
3116   }
3117   \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
3118   \seq_map_inline:Nn \l__stex_features_structure_field_seq {
3119     \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
3120     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3121       \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
3122       \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
3123         {!} \l_tmpa_tl
3124       \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3125         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
3126         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3127         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3128       }{
3129         \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
3130         \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
3131         \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
3132           \l_tmpa_tl
3133         \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
3134           \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
3135           \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
3136         }{
3137           \tl_clear:N \l_tmpb_tl
3138         }
3139       }
3140     }{
3141       \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
3142       \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
3143         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
3144         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
3145         \tl_clear:N \l_tmpa_tl
3146       }{
3147         % TODO throw error
3148       }
3149     }
3150     % \l_tmpa_str: name
3151     % \l_tmpa_tl: definiens
3152     % \l_tmpb_tl: notation
3153     \tl_if_empty:NT \l__stex_features_structure_field_str {
3154       % TODO throw error
3155     }
3156     \str_clear:N \l_tmpb_str
3157
3158     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3159     \seq_map_inline:Nn \l_tmpa_seq {
3160       \seq_set_split:Nnn \l_tmpb_seq ? { #####1 }

```

```

3161 \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
3162 \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
3163   \seq_map_break:n {
3164     \str_set:Nn \l_tmpb_str { ####1 }
3165   }
3166 }
3167 }
3168 \prop_get:cnN { l_stex_symdecl_ \l_tmpb_str _prop } {args}
3169 \l_tmpb_str
3170
3171 \tl_if_empty:NTF \l_tmpb_tl {
3172   \tl_if_empty:NF \l_tmpa_tl {
3173     \exp_args:Nx \use:n {
3174       \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
3175     }
3176   }
3177 }{
3178   \tl_if_empty:NTF \l_tmpa_tl {
3179     \exp_args:Nx \use:n {
3180       \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
3181     }
3182   }{
3183     \exp_args:Nx \use:n {
3184       \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
3185     }
3186     \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
3187   }
3188 }
3189 }
3190 % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
3191 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3192 % #3/\l__stex_features_structure_field_str
3193 % \par
3194 % \expandafter\present\csname
3195 %   l_stex_symdecl_
3196 % \prop_item:Nn \l_stex_current_module_prop {ns} ?
3197 % \prop_item:Nn \l_stex_current_module_prop {name} ?
3198 % #3/\l__stex_features_structure_field_str
3199 %   _prop
3200 % \endcsname
3201 }
3202
3203 \tl_clear:N \l__stex_features_structure_def_tl
3204
3205 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3206 \seq_map_inline:Nn \l_tmpa_seq {
3207   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3208   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3209   \exp_args:Nx \use:n {
3210     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
3211
3212     }
3213   }
3214 }

```

```

3215 \prop_if_exist:cF {
3216   l_stex_symdecl_
3217   \prop_item:Nn \l_stex_current_module_prop {ns} ?
3218   \prop_item:Nn \l_stex_current_module_prop {name} ?
3219   #3/\l_tmpa_str
3220   _prop
3221 }{
3222   \prop_get:cnN { l_stex_symdecl_ ##1 _prop } {args}
3223   \l_tmpb_str
3224   \exp_args:Nx \use:n {
3225     \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
3226   }
3227 }
3228 }
3229
3230 \symdecl*[type={\STEXsymbol{module-type}}{
3231   \_stex_term_math_oms:nnnn {
3232     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3233     \prop_item:Nn \l__stex_features_structure_prop {name}
3234     }{}{0}{}
3235   }{}{#3}
3236 }
3237 % TODO: -> sms file
3238
3239 \tl_set:cx{ #3 }{
3240   \stex_invoke_structure:nnn {
3241     \prop_item:Nn \l_stex_current_module_prop {ns} ?
3242     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3243   } {
3244     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3245     \prop_item:Nn \l__stex_features_structure_prop {name}
3246   }
3247 }
3248
3249 }

```

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

\stex_invoke_structure:nnn

```

3250 % #1: URI of the instance
3251 % #2: URI of the instantiated module
3252 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3253   \tl_if_empty:nTF{ #3 }{
3254     \prop_set_eq:Nc \l__stex_features_structure_prop {
3255       c_stex_feature_ #2 _prop
3256     }
3257     \tl_clear:N \l_tmpa_tl
3258     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3259     \seq_map_inline:Nn \l_tmpa_seq {
3260       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3261       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3262       \cs_if_exist:cT {
3263         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3264       }{

```

```

3265     \tl_if_empty:NF \l_tmpa_tl {
3266         \tl_put_right:Nn \l_tmpa_tl {,}
3267     }
3268     \tl_put_right:Nx \l_tmpa_tl {
3269         \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3270     }
3271 }
3272 }
3273 \exp_args:No \mathstruct \l_tmpa_tl
3274 }{
3275     \stex_invoke_symbol:n{#1/#3}
3276 }
3277 }

```

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

```

3278 </package>

```

Chapter 26

STEX -Statements Implementation

```
3279 <*package>
3280
3281 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3282
3283 \protected\def\ignorespacesandpars{
3284   \begingroup\catcode13=10\relax
3285   \@ifnextchar\par{
3286     \endgroup\expandafter\ignorespacesandpars\@gobble
3287   }{
3288     \endgroup
3289   }
3290 }
3291
3292 <@@=stex_statements>
3293
3294   Warnings and error messages
```

\titleemph

```
3294 \def\titleemph#1{\textbf{#1}}
```

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

26.1 Definitions

definiendum

```
3295 \keys_define:nn {stex / definiendum }{
3296   post      .tl_set:N      = \l__stex_statements_definiendum_post_tl,
3297   root      .str_set_x:N   = \l__stex_statements_definiendum_root_str,
3298   gfa       .str_set_x:N   = \l__stex_statements_definiendum_gfa_str
3299 }
3300 \cs_new_protected:Nn \__stex_statements_definiendum_args:n {
3301   \str_clear:N \l__stex_statements_definiendum_root_str
3302   \tl_clear:N \l__stex_statements_definiendum_post_tl
3303   \str_clear:N \l__stex_statements_definiendum_gfa_str
```



```

3304 \keys_set:nn { stex / definiendum }{ #1 }
3305 }
3306 \NewDocumentCommand \definiendum { 0{ } m m } {
3307   \__stex_statements_definiendum_args:n { #1 }
3308   \stex_get_symbol:n { #2 }
3309   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3310   \str_if_empty:NTF \l__stex_statements_definiendum_root_str {
3311     \tl_if_empty:NTF \l__stex_statements_definiendum_post_tl {
3312       \tl_set:Nn \l_tmpa_tl { #3 }
3313     } {
3314       \str_set:Nx \l__stex_statements_definiendum_root_str { #3 }
3315       \tl_set:Nn \l_tmpa_tl {
3316         \l__stex_statements_definiendum_root_str\l__stex_statements_definiendum_post_tl
3317       }
3318     }
3319   } {
3320     \tl_set:Nn \l_tmpa_tl { #3 }
3321   }
3322
3323   % TODO root
3324   \rustex_if:TF {
3325     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { \l_tmpa_tl }
3326   } {
3327     \exp_args:Nnx \defemph@uri { \l_tmpa_tl } { \l_stex_get_symbol_uri_str }
3328   }
3329 }
3330 \stex_deactivate_macro:Nn \definiendum {definition~environments}

```

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

definame

```

3331 \NewDocumentCommand \definame { 0{ } m } {
3332   \__stex_statements_definiendum_args:n { #1 }
3333   % TODO: root
3334   \stex_get_symbol:n { #2 }
3335   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3336   \str_set:Nx \l_tmpa_str {
3337     \prop_item:cn { l_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3338   }
3339   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3340   \rustex_if:TF {
3341     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3342       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3343     }
3344   } {
3345     \defemph@uri {
3346       \l_tmpa_str\l__stex_statements_definiendum_post_tl
3347     } { \l_stex_get_symbol_uri_str }
3348   }
3349 }
3350 \stex_deactivate_macro:Nn \definame {definition~environments}

```

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

sdefinition

```

3351
3352 \keys_define:nn {stex / sdefinition }{
3353   type      .str_set_x:N = \sdefinitiontype,
3354   id        .str_set_x:N = \sdefinitionid,
3355   title     .tl_set:N    = \sdefinitiontitle
3356 }
3357 \cs_new_protected:Nn \__stex_statements_sdefinition_args:n {
3358   \str_clear:N \sdefinitiontype
3359   \str_clear:N \sdefinitionid
3360   \tl_clear:N \sdefinitiontitle
3361   \keys_set:nn { stex / sdefinition }{ #1 }
3362 }
3363
3364 \NewDocumentEnvironment{sdefinition}{0{}}{
3365   \__stex_statements_sdefinition_args:n{ #1 }
3366   \stex_reactivate_macro:N \definiendum
3367   \stex_reactivate_macro:N \definame
3368   \stex_smsmode_set_codes:
3369   \clist_set:No \l_tmpa_clist \sdefinitiontype
3370   \tl_clear:N \l_tmpa_tl
3371   \clist_map_inline:Nn \l_tmpa_clist {
3372     \tl_if_exist:cT {__stex_statements_sdefinition_##1_start:}{
3373       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_start:}}
3374     }
3375   }
3376   \tl_if_empty:NTF \l_tmpa_tl {
3377     \__stex_statements_sdefinition_start:
3378   }{
3379     \l_tmpa_tl
3380   }
3381   \stex_ref_new_doc_target:n \sdefinitionid
3382   \stex_if_smsmode:F {
3383     \exp_args:Nnnx
3384     \begin{stex_annotate_env}{definition}{}
3385     \str_if_empty:NF \sdefinitiontype {
3386       \stex_annotate_invisible:nnn{type}{\sdefinitiontype}{}
3387     }
3388   }
3389   }{
3390     \stex_if_smsmode:F {
3391       \end{stex_annotate_env}
3392     }
3393     \clist_set:No \l_tmpa_clist \sdefinitiontype
3394     \tl_clear:N \l_tmpa_tl
3395     \clist_map_inline:Nn \l_tmpa_clist {
3396       \tl_if_exist:cT {__stex_statements_sdefinition_##1_end:}{
3397         \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sdefinition_##1_end:}}
3398       }
3399     }
3400     \tl_if_empty:NTF \l_tmpa_tl {
3401       \__stex_statements_sdefinition_end:
3402     }{
3403       \l_tmpa_tl

```

```

3404 }
3405 }

```

`\stexpatchdefinition`

```

3406 \cs_new_protected:Nn \__stex_statements_sdefinition_start: {
3407   \par\noindent\titleemph{Definition\tl_if_empty:NF \sdefinitiontitle {
3408     ~(\sdefinitiontitle)
3409   }~}
3410 }
3411 \cs_new_protected:Nn \__stex_statements_sdefinition_end: {\par\medskip}
3412
3413 \newcommand\stexpatchdefinition[3] [] {
3414   \str_set:Nx \l_tmpa_str{ #1 }
3415   \str_if_empty:NTF \l_tmpa_str {
3416     \tl_set:Nn \__stex_statements_sdefinition_start: { #2 }
3417     \tl_set:Nn \__stex_statements_sdefinition_end: { #3 }
3418   }{
3419     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_start:\endcsname{ #2 }
3420     \exp_after:wN \tl_set:Nn \csname __stex_statements_sdefinition_#1_end:\endcsname{ #3 }
3421   }
3422 }

```

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

`\inlinedef inline:`

```

3423 \NewDocumentCommand \inlinedef { m } {
3424   \begingroup
3425   \stex_reactivate_macro:N \definiendum
3426   \stex_reactivate_macro:N \definame
3427   \stex_ref_new_doc_target:n{}
3428   #1
3429   \endgroup
3430 }

```

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

26.2 Assertions

`sassertion`

```

3431
3432 \keys_define:nn {stex / sassertion }{
3433   type      .str_set_x:N = \sassertiontype,
3434   id        .str_set_x:N = \sassertionid,
3435   title     .tl_set:N     = \sassertiontitle ,
3436   name      .str_set_x:N = \sassertionname
3437 }
3438 \cs_new_protected:Nn \__stex_statements_sassertion_args:n {
3439   \str_clear:N \sassertiontype
3440   \str_clear:N \sassertionid
3441   \str_clear:N \sassertionname
3442   \tl_clear:N \sassertiontitle
3443   \keys_set:nn { stex / sassertion }{ #1 }
3444 }

```

```

3445
3446 \tl_new:N \g__stex_statements_aftergroup_tl
3447
3448 \NewDocumentEnvironment{sassertion}{0{}}{
3449   \__stex_statements_sassertion_args:n{ #1 }
3450   \stex_smsmode_set_codes:
3451   \clist_set:No \l_tmpa_clist \sassertiontype
3452   \tl_clear:N \l_tmpa_tl
3453   \clist_map_inline:Nn \l_tmpa_clist {
3454     \tl_if_exist:cT {\__stex_statements_sassertion_##1_start:}{
3455       \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_start:}}
3456     }
3457   }
3458   \tl_if_empty:NTF \l_tmpa_tl {
3459     \__stex_statements_sassertion_start:
3460   }{
3461     \l_tmpa_tl
3462   }
3463   \stex_ref_new_doc_target:n \sassertionid
3464   \stex_if_smsmode:F {
3465     \exp_args:Nnnx
3466     \begin{stex_annotate_env}{assertion}{}
3467     \str_if_empty:NF \sassertiontype {
3468       \stex_annotate_invisible:nnn{type}{\sassertiontype}{}
3469     }
3470   }
3471   }{
3472     \stex_if_smsmode:F {
3473       \end{stex_annotate_env}
3474     }
3475     \clist_set:No \l_tmpa_clist \sassertiontype
3476     \tl_clear:N \l_tmpa_tl
3477     \clist_map_inline:Nn \l_tmpa_clist {
3478       \tl_if_exist:cT {\__stex_statements_sassertion_##1_end:}{
3479         \tl_set:Nn \l_tmpa_tl {\use:c{\__stex_statements_sassertion_##1_end:}}
3480       }
3481     }
3482     \tl_if_empty:NTF \l_tmpa_tl {
3483       \__stex_statements_sassertion_end:
3484     }{
3485       \l_tmpa_tl
3486     }
3487     \str_if_empty:NF \sassertionname {
3488       \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3489         \symdecl*{\sassertionname}
3490       }
3491       \aftergroup\g__stex_statements_aftergroup_tl
3492     }
3493   }

```

\stexpatchassertion

```

3494
3495 \cs_new_protected:Nn \__stex_statements_sassertion_start: {
3496   \par\noindent\titllemph{Assertion~\tl_if_empty:NF \sassertiontitle {

```

```

3497     (\sassertiontitle)
3498   }~}
3499 }
3500 \cs_new_protected:Nn \__stex_statements_sassertion_end: {\par\medskip}
3501
3502 \newcommand\stexpatchassertion[3] [] {
3503   \str_set:Nx \l_tmpa_str{ #1 }
3504   \str_if_empty:NTF \l_tmpa_str {
3505     \tl_set:Nn \__stex_statements_sassertion_start: { #2 }
3506     \tl_set:Nn \__stex_statements_sassertion_end: { #3 }
3507   }{
3508     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_start:\endcsname{ #2
3509     \exp_after:wN \tl_set:Nn \csname __stex_statements_sassertion_#1_end:\endcsname{ #3 }
3510   }
3511 }

```

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

\inlineass inline:

```

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

```

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

26.3 Examples

sexample

```

3518
3519 \keys_define:nn {stex / sexample }{
3520   type      .str_set_x:N = \exampletype,
3521   id        .str_set_x:N = \sexampleid,
3522   title     .tl_set:N = \sexampletitle,
3523   for       .clist_set:N = \sexamplefor,
3524 }
3525 \cs_new_protected:Nn \__stex_statements_sexample_args:n {
3526   \str_clear:N \sexampletype
3527   \str_clear:N \sexampleid
3528   \tl_clear:N \sexampletitle
3529   \clist_clear:N \sexamplefor
3530   \keys_set:nn { stex / sexample }{ #1 }
3531 }
3532
3533 \NewDocumentEnvironment{sexample}{0{}}{
3534   \__stex_statements_sexample_args:n{ #1 }
3535   \stex_smsmode_set_codes:
3536   \clist_set:Nn \l_tmpa_clist \sexampletype
3537   \tl_clear:N \l_tmpa_tl
3538   \clist_map_inline:Nn \l_tmpa_clist {
3539     \tl_if_exist:cT {\__stex_statements_sexample_##1_start:}{

```

```

3540     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_start:}}
3541   }
3542 }
3543 \tl_if_empty:NTF \l_tmpa_tl {
3544   \__stex_statements_sexample_start:
3545 }{
3546   \l_tmpa_tl
3547 }
3548 \stex_ref_new_doc_target:n \sexampleid
3549 \stex_if_smsmode:F {
3550   \seq_clear:N \l_tmpa_seq
3551   \clist_map_inline:Nn \sexamplefor {
3552     \str_if_eq:nnF{ ##1 }{{}{
3553       \stex_get_symbol:n { ##1 }
3554       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq {
3555         \l_stex_get_symbol_uri_str
3556       }
3557     }
3558   }
3559   \exp_args:Nnnx
3560   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpa_seq {,}}
3561   \str_if_empty:NF \sexamplotype {
3562     \stex_annotate_invisible:nnn{type}{\sexampletype}{}
3563   }
3564 }
3565 }{
3566   \stex_if_smsmode:F {
3567     \end{stex_annotate_env}
3568   }
3569   \clist_set:Nn \l_tmpa_clist \sexamplotype
3570   \tl_clear:N \l_tmpa_tl
3571   \clist_map_inline:Nn \l_tmpa_clist {
3572     \tl_if_exist:cT {__stex_statements_sexample_##1_end:}{
3573       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sexample_##1_end:}}
3574     }
3575   }
3576   \tl_if_empty:NTF \l_tmpa_tl {
3577     \__stex_statements_sexample_end:
3578   }{
3579     \l_tmpa_tl
3580   }
3581 }

```

\stexpatchexample

```

3582
3583 \cs_new_protected:Nn \__stex_statements_sexample_start: {
3584   \par\noindent\titleemph{Example~\tl_if_empty:NF \sexamplotype {
3585     (\sexamplotype)
3586   }~}
3587 }
3588 \cs_new_protected:Nn \__stex_statements_sexample_end: {\par\medskip}
3589
3590 \newcommand\stexpatchexample[3][ ] {
3591   \str_set:Nx \l_tmpa_str{ #1 }

```

```

3592 \str_if_empty:NTF \l_tmpa_str {
3593   \tl_set:Nn \__stex_statements_sexample_start: { #2 }
3594   \tl_set:Nn \__stex_statements_sexample_end: { #3 }
3595 }{
3596   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_start:\endcsname{ #2 }
3597   \exp_after:wN \tl_set:Nn \csname __stex_statements_sexample_#1_end:\endcsname{ #3 }
3598 }
3599 }

```

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

`\inlineex` inline:

```

3600 \NewDocumentCommand \inlineex { m } {
3601   \beginngroup
3602   \stex_ref_new_doc_target:n{
3603     #1
3604   \endgroup
3605 }

```

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

26.4 Logical Paragraphs

`sparagraph`

```

3606 \keys_define:nn { stex / sparagraph } {
3607   id      .str_set_x:N = \sparagraphid ,
3608   title   .tl_set:N    = \l_stex_sparagraph_title_tl ,
3609   type    .str_set_x:N = \sparagraphtype ,
3610   for     .str_set_x:N = \sparagraphfor ,
3611   from    .tl_set_x:N  = \sparagraphfrom ,
3612   start   .tl_set:N    = \l_stex_sparagraph_start_tl ,
3613   name    .str_set:N    = \sparagraphname
3614 }
3615
3616 \cs_new_protected:Nn \stex_sparagraph_args:n {
3617   \tl_clear:N \l_stex_sparagraph_title_tl
3618   \tl_clear:N \sparagraphfrom
3619   \tl_clear:N \l_stex_sparagraph_start_tl
3620   \str_clear:N \sparagraphid
3621   \str_clear:N \sparagraphtype
3622   \str_clear:N \sparagraphfor
3623   \str_clear:N \sparagraphname
3624   \keys_set:nn { stex / sparagraph }{ #1 }
3625 }
3626 \newif\if@in@omtext\@in@omtextfalse
3627
3628 \NewDocumentEnvironment {sparagraph} { 0{} } {
3629   \stex_sparagraph_args:n { #1 }
3630   \tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3631     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_title_tl
3632   }{
3633     \tl_set_eq:NN \sparagraphtitle \l_stex_sparagraph_start_tl
3634   }

```

```

3635 \@in@omtexttrue
3636 \stex_smsmode_set_codes:
3637 \clist_set:No \l_tmpa_clist \sparagraphtype
3638 \tl_clear:N \l_tmpa_tl
3639 \clist_map_inline:Nn \l_tmpa_clist {
3640   \tl_if_exist:cT {__stex_statements_sparagraph_##1_start:}{
3641     \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_start:}}
3642   }
3643 }
3644 \tl_if_empty:NTF \l_tmpa_tl {
3645   \__stex_statements_sparagraph_start:
3646 }{
3647   \l_tmpa_tl
3648 }
3649 \stex_ref_new_doc_target:n \sparagraphid
3650 \stex_if_smsmode:F {
3651   \exp_args:Nnnx
3652   \begin{stex_annotate_env}{paragraph}{}
3653   \str_if_empty:NF \sparagraphtype {
3654     \stex_annotate_invisible:nnn{type}{\sparagraphtype}{}
3655   }
3656 }
3657 \ignorespacesandpars
3658 }{
3659   \stex_if_smsmode:F {
3660     \end{stex_annotate_env}
3661   }
3662   \clist_set:No \l_tmpa_clist \sparagraphtype
3663   \tl_clear:N \l_tmpa_tl
3664   \clist_map_inline:Nn \l_tmpa_clist {
3665     \tl_if_exist:cT {__stex_statements_sparagraph_##1_end:}{
3666       \tl_set:Nn \l_tmpa_tl {\use:c{__stex_statements_sparagraph_##1_end:}}
3667     }
3668   }
3669   \tl_if_empty:NTF \l_tmpa_tl {
3670     \__stex_statements_sparagraph_end:
3671   }{
3672     \l_tmpa_tl
3673   }
3674   \str_if_empty:NF \sparagraphname {
3675     \tl_gset:Nx \g__stex_statements_aftergroup_tl {
3676       \symdecl*{\sparagraphname}
3677     }
3678     \aftergroup\g__stex_statements_aftergroup_tl
3679   }
3680 }

```

\stexpatchparagraph

```

3681
3682 \cs_new_protected:Nn \__stex_statements_sparagraph_start: {
3683   \par\noindent\tl_if_empty:NTF \l_stex_sparagraph_start_tl {
3684     \tl_if_empty:NF \l_stex_sparagraph_title_tl {
3685       \titleemph{\l_stex_sparagraph_title_tl}:~
3686     }

```



```

3687   }{
3688     \titleemph{\l_stex_sparagraph_start_tl}~
3689   }
3690 }
3691 \cs_new_protected:Nn \__stex_statements_sparagraph_end: {\par\medskip}
3692
3693 \newcommand\stexpatchparagraph[3] [] {
3694   \str_set:Nx \l_tmpa_str{ #1 }
3695   \str_if_empty:NTF \l_tmpa_str {
3696     \tl_set:Nn \__stex_statements_sparagraph_start: { #2 }
3697     \tl_set:Nn \__stex_statements_sparagraph_end: { #3 }
3698   }{
3699     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_start:\endcsname{ #2
3700     \exp_after:wN \tl_set:Nn \csname __stex_statements_sparagraph_#1_end:\endcsname{ #3 }
3701   }
3702 }

```

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

symboldoc

```

3703 \NewDocumentEnvironment{symboldoc}{ m }{
3704   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3705   \seq_clear:N \l_tmpb_seq
3706   \seq_map_inline:Nn \l_tmpa_seq {
3707     \str_if_eq:nnF{ ##1 }{}{
3708       \stex_get_symbol:n { ##1 }
3709       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3710         \l_stex_get_symbol_uri_str
3711       }
3712     }
3713   }
3714   \par
3715   \exp_args:Nnnx
3716   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3717 }{
3718   \end{stex_annotate_env}
3719 }
3720 \</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

```
3721 <*package>
3722 <@@=stex_sproof>
3723
3724 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3725
```

27.2 Proofs

We first define some keys for the proof environment.

```
3726 \keys_define:nn { stex / spf } {
3727   id          .str_set:N = \l__stex_sproof_spf_id_str,
3728   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3729   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3730   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3731   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3732   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3733   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3734   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3735   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3736   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3737 }
3738 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3739   \str_clear:N \l__stex_sproof_spf_id_str
3740   \tl_clear:N \l__stex_sproof_spf_display_tl
3741   \tl_clear:N \l__stex_sproof_spf_for_tl
3742   \tl_clear:N \l__stex_sproof_spf_from_tl
3743   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3744   \tl_clear:N \l__stex_sproof_spf_type_tl
3745   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹⁰EDNOTE: need an implementation for L^AT_EX_ML

```

3746 \tl_clear:N \l__stex_sproof_spf_continues_tl
3747 \tl_clear:N \l__stex_sproof_spf_functions_tl
3748 \tl_clear:N \l__stex_sproof_spf_method_tl
3749 \keys_set:nn { stex / spf }{ #1 }
3750 }

```

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

```

3751 \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.

```

3752 \newcount\count_ten
3753 \newenvironment{pst@with@label}[1]{
3754   \edef\pst@label{#1}
3755   \advance\count_ten by 1\relax
3756   \count_ten=1
3757 }{
3758   \advance\count_ten by -1\relax
3759 }

```

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

```

3760 \def\the@pst@label{
3761   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3762 }

```

(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.

```

3763 \keys_define:nn { stex / pstlabel }{
3764   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3765   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3766   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3767 }
3768 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

3769 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3770 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3771 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3772 }
3773 \__stex_sproof_pstlabel_args:n {}
3774 \newcommand\setpstlabelstyle[1]{
3775   \__stex_sproof_pstlabel_args:n {#1}
3776 }
3777 \newcommand\setpstlabelstyledefault{%
3778   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3779 }

```

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

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

```

3780 \ExplSyntaxOff
3781 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3782 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3783 \def\pst@make@label@short#1#2{#2}
3784 \def\pst@make@label@empty#1#2{}
3785 \ExplSyntaxOn
3786 \def\pstlabelstyle#1{%
3787   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3788 }%
3789 \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.

```

3790 \def\next@pst@label{%
3791   \global\advance\count\count10 by 1%
3792 }%

```

(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

```

3793 \def\sproof@box{
3794   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3795 }
3796 \def\spf@proofend{\sproof@box}
3797 \def\sproofend{
3798   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3799     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3800   }
3801 }
3802 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3803 \def\spf@proofsketch@kw{Proof Sketch}
3804 \def\spf@proof@kw{Proof}
3805 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3806 \cs_if_exist:NT \bbl@loaded {
3807   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3808   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3809     \input{proof-ngerman.lda}
3810   }
3811   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3812     \input{proof-finnish.lda}
3813   }
3814   \clist_if_in:NnT \l_tmpa_clist {french}{
3815     \input{proof-french.lda}
3816   }
3817   \clist_if_in:NnT \l_tmpa_clist {russian}{
3818     \input{proof-russian.lda}
3819   }
3820 }
3821

```

`spfsketch`

```

3822 \newcommand\spfsketch[2][]{
3823   \__stex_sproof_spf_args:n{#1}
3824   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3825     \titleemph{
3826       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3827         \spf@proofsketch@kw
3828       }{
3829         \l__stex_sproof_spf_type_tl
3830       }
3831     }:
3832   }
3833   {-#2}
3834   %\sref@label@id{this \ifx\spf@type\empty\spf@proofsketch@kw\else\spf@type\fi}
3835   \sproofend
3836 }

```

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

EdN:11
EdN:12

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

```

3837 \newenvironment{spfeq}[2][]{
3838   \__stex_sproof_spf_args:n{#1}
3839   %\sref@target
3840   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3841     \titleemph{
3842       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3843         \spf@proof@kw
3844       }{
3845         \l__stex_sproof_spf_type_tl
3846       }
3847     }:

```

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

¹²EDNOTE: document above

```

3848 }
3849 {~#2}
3850 \begin{displaymath}\begin{array}{rcll}
3851 }{
3852 \end{array}\end{displaymath}
3853 }

```

(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.

```

3854 \newenvironment{spf@proof}[2][]{
3855   \__stex_sproof_spf_args:n{#1}
3856   %\sref@target
3857   \count_ten=10
3858   \par\noindent
3859   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3860     \titleemph{
3861       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3862         \spf@proof@kw
3863       }{
3864         \l__stex_sproof_spf_type_tl
3865       }
3866     }:
3867   }
3868   {~#2}
3869   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
3870   \def\pst@label{}
3871   \newcount\pst@count% initialize the labeling mechanism
3872   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3873   }{
3874     \end{pst@with@label}\end{description}
3875   }
3876 \newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3877 \newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3878 \newcommand\spfidea[2][]{
3879   \__stex_sproof_spf_args:n{#1}
3880   \titleemph{
3881     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3882       \l__stex_sproof_spf_type_tl
3883     }:
3884   }~#2
3885   \sproofend
3886 }

```

(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

```

3887 \newenvironment{spfstep}[1][]{
3888   \_stex_sproof_spf_args:n{#1}
3889   \@in@omtexttrue
3890   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3891     \item[\the@pst@label]
3892   }
3893   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3894     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3895   }
3896   %\sref@label@id{\pst@label}
3897   \ignorespacesandpars
3898 }{
3899   \next@pst@label\ignorespacesandpars
3900 }

```

sproofcomment

```

3901 \newenvironment{sproofcomment}[1][]{
3902   \_stex_sproof_spf_args:n{#1}
3903   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3904     \item[\the@pst@label]
3905   }
3906 }{
3907   \next@pst@label
3908 }

```

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.

```

3909 \newenvironment{subproof}[2][]{
3910   \_stex_sproof_spf_args:n{#1}
3911   \def\@test{#2}
3912   \ifx\@test\empty\else
3913     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3914       \item[\the@pst@label]
3915     }{#2}
3916   \fi
3917   \begin{pst@with@label}{\pst@label,\number\count_ten}
3918 }{
3919   \end{pst@with@label}\next@pst@label
3920 }

```

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

```

3921 \newenvironment{spfcases}[2][]{
3922   \def\@test{#1}
3923   \ifx\@test\empty
3924     \begin{subproof}[method=by-cases]{#2}
3925   \else
3926     \begin{subproof}[#1,method=by-cases]{#2}
3927   \fi
3928 }{

```

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

```

3929 \end{subproof}
3930 }

```

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

```

3931 \newenvironment{spfcase}[2] [] {
3932   \__stex_sproof_spf_args:n{#1}
3933   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3934     \item[\the@pst@label]
3935   }
3936   \def\@test{#2}
3937   \ifx\@test\@empty
3938   \else
3939     {\titleemph{#2}:~}
3940   \fi
3941   \begin{pst@with@label}{\pst@label,\number\count_ten}
3942   }{
3943     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3944       \sproofend
3945     }
3946     \end{pst@with@label}
3947     \next@pst@label
3948   }

```

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

```

3949 \newcommand\spfcasesketch[3] [] {
3950   \__stex_sproof_spf_args:n{#1}
3951   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3952     \item[\the@pst@label]
3953   }
3954   \def\@test{#2}
3955   \ifx\@test\@empty
3956   \else
3957     {\titleemph{#2}:~}
3958   \fi#3
3959   \next@pst@label
3960 }%

```

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.

```

3961 \keys_define:nn { stex / just }{
3962   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3963   method      .tl_set:N   = \l__stex_sproof_just_method_tl,
3964   premises    .tl_set:N   = \l__stex_sproof_just_premises_tl,
3965   args        .tl_set:N   = \l__stex_sproof_just_args_tl
3966 }

```

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

```
3967 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3968 \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.

```
3969 \newcommand\justarg[2] [] {#2}
```

```
3970 \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

```
3971 <*package>
3972
3973 %%%%%%%%%% others.dtx %%%%%%%%%%
3974
3975 <@@=stex_others>
    Warnings and error messages
3976 % None

\MSC Math subject classifier

3977 \NewDocumentCommand \MSC {m} {
3978 % TODO
3979 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3980 \@ifpackageloaded{tikzinput}{
3981 \RequirePackage{stex-tikzinput}
3982 }{}
3983 </package>
```

Chapter 29

STEX -Metatheory Implementation

```
3984 <*package>
3985 <@@=stex_modules>
3986
3987 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3988
3989 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3990 \begingroup
3991 \stex_module_setup:nn{
3992   ns=\c_stex_metatheory_ns_str,
3993   meta=NONE
3994 }{Metatheory}
3995 \stex_reactivate_macro:N \symdecl
3996 \stex_reactivate_macro:N \notation
3997 \stex_reactivate_macro:N \symdef
3998 \ExplSyntaxOff
3999 \csname stex_suppress_html:n\endcsname{
4000   % is-a (a:A, a \in A, a is an A, etc.)
4001   \symdecl[args=ai]{isa}
4002   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
4003   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
4004   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
4005
4006   % bind (\forall, \Pi, \lambda etc.)
4007   \symdecl[args=Bi]{bind}
4008   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
4009   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
4010   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; )}{#1 \comp, #2}
4011
4012   % dummy variable
4013   \symdecl{dummyvar}
4014   \notation[underscore]{dummyvar}{\comp\_}
4015   \notation[dot]{dummyvar}{\comp\cdot}
4016   \notation[dash]{dummyvar}{\comp{\rm --}}
4017
4018   %fromto (function space, Hom-set, implication etc.)
```

```

4019 \symdecl[args=ai]{fromto}
4020 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
4021 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
4022
4023 % mapto (lambda etc.)
4024 %\symdecl[args=Bi]{mapto}
4025 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
4026 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
4027 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
4028
4029 % function/operator application
4030 \symdecl[args=ia]{apply}
4031 \notation[prec=0;0x\infpres,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
4032 \notation[prec=0;0x\infpres,lambda]{apply}{#1 \; #2 }{#1 \; #2}
4033
4034 % ‘type’ of all collections (sets, classes, types, kinds)
4035 \symdecl{collection}
4036 \notation[U]{collection}{\comp{\mathcal{U}}}
4037 \notation[set]{collection}{\comp{\textsf{Set}}}
4038
4039 % sequences
4040 \symdecl[args=1]{seqtype}
4041 \notation[kleene]{seqtype}{#1^{\comp\ast}}
4042
4043 \symdef[args=2,li,prec=nobrackets]{sequence-index}{#1_{#2}}
4044 \notation[ui,prec=nobrackets]{sequence-index}{#1^{#2}}
4045
4046 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses},#1_{#3}}
4047 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses},#1^{#3}}
4048 % ^ superceded by \aseqfromto and \livar/\uivar
4049
4050 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
4051 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses},#2}{#1\comp,#2}
4052 \symdef[args=aui,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses},#2\comp{\,\ellipses},#3}
4053
4054 % letin (‘let’, local definitions, variable substitution)
4055 \symdecl[args=bii]{letin}
4056 \notation[let]{letin}{\comp{\rm let}}{#1\comp{=}#2\; \comp{\rm in}}{#3}
4057 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
4058 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
4059
4060 % structures
4061 \symdecl*[args=1]{module-type}
4062 \notation{module-type}{\mathtt{MOD} #1}
4063 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
4064 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
4065
4066 }
4067 \ExplSyntaxOn
4068 \stex_add_to_current_module:n{
4069   \let\nappa\apply
4070   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
4071   \def\nappui#1#2#3#4{\apply{#1}{\nasequi{#2}{#3}{#4}}}
4072   \def\livar{\csname sequence-index\endcsname[li]}

```

```

4073 \def\uivar{\csname sequence-index\endcsname[ui]}
4074 \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
4075 \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
4076 \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
4077 }
4078 \__stex_modules_end_module:
4079 \endgroup
4080 \</package>

```

Chapter 30

Tikzinput Implementation

```
4081 <*package>
4082
4083 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
4084
4085 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
4086 \RequirePackage{l3keys2e}
4087
4088 \keys_define:nn { tikzinput } {
4089   image .bool_set:N = \c_tikzinput_image_bool,
4090   image .default:n = false ,
4091   unknown .code:n = {}
4092 }
4093
4094 \ProcessKeysOptions { tikzinput }
4095
4096 \bool_if:NTF \c_tikzinput_image_bool {
4097   \RequirePackage{graphicx}
4098
4099   \providecommand\usetikzlibrary[]{}
4100   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
4101 }{
4102   \RequirePackage{tikz}
4103   \RequirePackage{standalone}
4104
4105   \newcommand \tikzinput [2] [] {
4106     \setkeys{Gin}{#1}
4107     \ifx \Gin@ewidth \Gin@exclamation
4108       \ifx \Gin@eheight \Gin@exclamation
4109         \input { #2 }
4110       \else
4111         \resizebox{!}{ \Gin@eheight }{
4112           \input { #2 }
4113         }
4114       \fi
4115     \else
4116       \ifx \Gin@eheight \Gin@exclamation
4117         \resizebox{ \Gin@ewidth }{!}{
4118           \input { #2 }
```

```

4119     }
4120     \else
4121         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
4122             \input { #2 }
4123         }
4124     \fi
4125 \fi
4126 }
4127 }
4128
4129 \newcommand \ctikzinput [2] [] {
4130     \begin{center}
4131         \tikzinput [1] {#2}
4132     \end{center}
4133 }
4134
4135 \@ifpackageloaded{stex}{
4136     \RequirePackage{stex-tikzinput}
4137 }{}
4138
4139 </package>
4140 <*stex>
4141 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
4142 \RequirePackage{stex}
4143 \RequirePackage{tikzinput}
4144
4145 \newcommand\mhtikzinput [2] [] {%
4146     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
4147     \stex_in_repository:nn\Gin@mhrepos{
4148         \tikzinput[#1]{\mhpath{##1}{#2}}
4149     }
4150 }
4151 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
4152 </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.

```
4153 \*cls)
4154 \@@=document_structure)
4155 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
4156 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

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`

```
4157 \keys_define:nn{ document-structure / pkg }{
4158   class      .str_set_x:N = \c_document_structure_class_str,
4159   minimal    .bool_set:N = \c_document_structure_minimal_bool,
4160   report     .code:n      = {
4161     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
4162     \str_set:Nn \c_document_structure_class_str {report}
4163   },
4164   book       .code:n      = {
4165     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
4166     \str_set:Nn \c_document_structure_class_str {book}
4167   },
4168   bookpart   .code:n      = {
4169     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
4170     \str_set:Nn \c_document_structure_class_str {book}
4171     \str_set:Nn \c_document_structure_topsect_str {chapter}
4172   },
```



```

4173 docopt      .str_set_x:N = \c_document_structure_docopt_str,
4174 unknown     .code:n      = {
4175   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
4176 }
4177 }
4178 \ProcessKeysOptions{ document-structure / pkg }
4179 \str_if_empty:NT \c_document_structure_class_str {
4180   \str_set:Nn \c_document_structure_class_str {article}
4181 }
4182 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
4183   {\c_document_structure_class_str}
4184

```

31.3 Beefing up the document environment

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

```

4185 \RequirePackage{omdoc}
4186 \bool_if:NF \c_document_structure_minimal_bool {
4187   \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

```

4188 \keys_define:nn { document-structure / document }{
4189   id .str_set_x:N = \c_document_structure_document_id_str
4190 }
4191 \let\__document_structure_orig_document=\document
4192 \renewcommand{\document}[1][]{
4193   \keys_set:nn{ document-structure / document }{ #1 }
4194   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
4195   \__document_structure_orig_document
4196 }

```

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

```

4197 }
4198 \</cls>

```

31.4 Implementation: OMDoc Package

```

4199 \<*package>
4200 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
4201 \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

```

4202
4203 \keys_define:nn{ document-structure / pkg }{
4204   class      .str_set_x:N = \c_document_structure_class_str,
4205   topsect    .str_set_x:N = \c_document_structure_topsect_str,
4206   % showignores .bool_set:N = \c_document_structure_showignores_bool,
4207 }
4208 \ProcessKeysOptions{ document-structure / pkg }
4209 \str_if_empty:NT \c_document_structure_class_str {
4210   \str_set:Nn \c_document_structure_class_str {article}
4211 }
4212 \str_if_empty:NT \c_document_structure_topsect_str {
4213   \str_set:Nn \c_document_structure_topsect_str {section}
4214 }

```

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

```

4215 \RequirePackage{xspace}
4216 \RequirePackage{comment}
4217 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

4218 \@ifpackageloaded{babel}{
4219   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4220   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4221     \input{omdoc-ngerman.ldf}
4222   }
4223 }{}
4224 %\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.

```

4225 \int_new:N \l_document_structure_section_level_int
4226 \str_case:VnF \c_document_structure_topsect_str {
4227   {part}{
4228     \int_set:Nn \l_document_structure_section_level_int {0}
4229   }
4230   {chapter}{
4231     \int_set:Nn \l_document_structure_section_level_int {1}
4232   }
4233 }{
4234   \str_case:VnF \c_document_structure_class_str {
4235     {book}{
4236       \int_set:Nn \l_document_structure_section_level_int {0}
4237     }
4238     {report}{
4239       \int_set:Nn \l_document_structure_section_level_int {0}
4240     }
4241   }{
4242     \int_set:Nn \l_document_structure_section_level_int {2}
4243   }
4244 }

```

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

```
4245 \def\current@section@level{document}%
4246 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
4247 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
4248 \cs_new_protected:Npn \skipomgroup {
4249   \ifcase\l_document_structure_section_level_int
4250   \or\stepcounter{part}
4251   \or\stepcounter{chapter}
4252   \or\stepcounter{section}
4253   \or\stepcounter{subsection}
4254   \or\stepcounter{subsubsection}
4255   \or\stepcounter{paragraph}
4256   \or\stepcounter{subparagraph}
4257   \fi
4258 }
```

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

`blindomgroup`

```
4259 \newcommand\at@begin@blindomgroup[1]{%
4260 \newenvironment{blindomgroup}
4261 {
4262   \int_incr:N\l_document_structure_section_level_int
4263   \at@begin@blindomgroup\l_document_structure_section_level_int
4264 }{}}
```

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

```
4265 \newcommand\omgroup@nonum[2]{
4266   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
4267   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
4268 }
```

(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.

```
4269 \newcommand\omgroup@num[2]{
```

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

```

4270 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
4271   \@nameuse{#1}{#2}
4272 }{
4273   \cs_if_exist:NTF\rdfmata@sectioning{
4274     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
4275   }{
4276     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
4277   }
4278 }
4279 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
4280 }

```

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

omgroup

```

4281 \keys_define:nn { document-structure / omgroup }{
4282   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
4283   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
4284   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
4285   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
4286   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
4287   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
4288   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
4289   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
4290   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
4291   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
4292 }
4293 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
4294   \str_clear:N \l__document_structure_omgroup_id_str
4295   \str_clear:N \l__document_structure_omgroup_date_str
4296   \clist_clear:N \l__document_structure_omgroup_creators_clist
4297   \clist_clear:N \l__document_structure_omgroup_contributors_clist
4298   \tl_clear:N \l__document_structure_omgroup_srccite_tl
4299   \tl_clear:N \l__document_structure_omgroup_type_tl
4300   \tl_clear:N \l__document_structure_omgroup_short_tl
4301   \tl_clear:N \l__document_structure_omgroup_display_tl
4302   \tl_clear:N \l__document_structure_omgroup_intro_tl
4303   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
4304   \keys_set:nn { document-structure / omgroup } { #1 }
4305 }

```

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.

```

4306 \newif\if@mainmatter\@mainmattertrue
4307 \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.

```

4308 \keys_define:nn { document-structure / sectioning }{
4309   name .str_set_x:N = \l__document_structure_sect_name_str ,
4310   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
4311   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
4312   num .bool_set:N = \l__document_structure_sect_num_bool ,
4313 }

```

```

4314 \cs_new_protected:Nn \__document_structure_sect_args:n {
4315   \str_clear:N \l__document_structure_sect_name_str
4316   \str_clear:N \l__document_structure_sect_ref_str
4317   \bool_set_false:N \l__document_structure_sect_clear_bool
4318   \bool_set_false:N \l__document_structure_sect_num_bool
4319   \keys_set:nn { document-structure / sectioning } { #1 }
4320 }
4321 \newcommand\omdoc@sectioning[3][]{
4322   \__document_structure_sect_args:n {#1}
4323   \let\omdoc@sect@name\l__document_structure_sect_name_str
4324   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
4325   \if@mainmatter% numbering not overridden by frontmatter, etc.
4326     \bool_if:NTF \l__document_structure_sect_num_bool {
4327       \omgroup@num{#2}{#3}
4328     }{
4329       \omgroup@nonum{#2}{#3}
4330     }
4331     \def\current@section@level{\omdoc@sect@name}
4332   \else
4333     \omgroup@nonum{#2}{#3}
4334   \fi
4335 }% 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.

```

4336 \newcommand\omgroup@redefine@addtocontents[1]{%
4337   %\edef\__document_structureimport{#1}%
4338   %\@for\@I:=\__document_structureimport\do{%
4339     %\edef\@path{\csname module@\@I @path\endcsname}%
4340     %\@ifundefined{tf@toc}\relax%
4341     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
4342   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
4343   %\def\addcontentsline##1##2##3{%
4344     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4345   %\else% hyperref.sty not loaded
4346   %\def\addcontentsline##1##2##3{%
4347     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
4348   %\fi
4349 }% 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.

```

4350 \int_new:N \l_document_structure_omgroup_level_int
4351 \newenvironment{omgroup}[2][]{% keys, title
4352 {
4353   \__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.

```

4354 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
4355   \omgroup@redefine@addtocontents{
4356     %\@ifundefined{module@id}\used@modules%
4357     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

4358     }
4359 }

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

4360 \int_incr:N \l_document_structure_omgroup_level_int
4361 \int_incr:N \l_document_structure_section_level_int
4362 \ifcase\l_document_structure_section_level_int
4363   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
4364   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
4365   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
4366   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
4367   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
4368   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
4369   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
4370 \fi
4371 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
4372 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
4373 }% for customization
4374 {}

```

and finally, we localize the sections

```

4375 \newcommand\omdoc@part@kw{Part}
4376 \newcommand\omdoc@chapter@kw{Chapter}
4377 \newcommand\omdoc@section@kw{Section}
4378 \newcommand\omdoc@subsection@kw{Subsection}
4379 \newcommand\omdoc@subsubsection@kw{Subsubsection}
4380 \newcommand\omdoc@paragraph@kw{paragraph}
4381 \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`

```

4382 \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).

```

4383 \cs_if_exist:NTF\frontmatter{
4384   \let\__document_structure_orig_frontmatter\frontmatter
4385   \let\frontmatter\relax
4386 }{
4387   \tl_set:Nn\__document_structure_orig_frontmatter{
4388     \clearpage
4389     \@mainmatterfalse
4390     \pagenumbering{roman}
4391   }
4392 }
4393 \cs_if_exist:NTF\backmatter{

```

```

4394 \let\__document_structure_orig_backmatter\backmatter
4395 \let\backmatter\relax
4396 }{
4397 \tl_set:Nn\__document_structure_orig_backmatter{
4398 \clearpage
4399 \@mainmatterfalse
4400 \pagenumbering{roman}
4401 }
4402 }

```

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.

```

4403 \newenvironment{frontmatter}{
4404 \__document_structure_orig_frontmatter
4405 }{
4406 \cs_if_exist:NTF\mainmatter{
4407 \mainmatter
4408 }{
4409 \clearpage
4410 \@mainmattertrue
4411 \pagenumbering{arabic}
4412 }
4413 }

```

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

```

4414 \newenvironment{backmatter}{
4415 \__document_structure_orig_backmatter
4416 }{
4417 \cs_if_exist:NTF\mainmatter{
4418 \mainmatter
4419 }{
4420 \clearpage
4421 \@mainmattertrue
4422 \pagenumbering{arabic}
4423 }
4424 }

```

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

```

4425 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4426 \def \c__document_structure_document_str{document}
4427 \newcommand\afterprematurestop{}
4428 \def\prematurestop@endomgroup{
4429 \unless\ifx\@currenvir\c__document_structure_document_str
4430 \expandafter\expandafter\expandafter\end\expandafter\expandafter\expandafter{\expandafter
4431 \expandafter\prematurestop@endomgroup
4432 \fi
4433 }
4434 \providecommand\prematurestop{

```

```

4435 \message{Stopping~sTeX~processing~prematurely}
4436 \prematurestop@endomgroup
4437 \afterprematurestop
4438 \end{document}
4439 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

4440 \RequirePackage{etoolbox}
4441 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4442 \newrobustcmd\useSGvar[1]{%
4443 \@ifundefined{sTeX@Gvar@#1}
4444 {\PackageError{omdoc}
4445 {The sTeX Global variable #1 is undefined}
4446 {set it with \protect\setSGvar}}
4447 \@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.

```

4448 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4449 \@ifundefined{sTeX@Gvar@#1}
4450 {\PackageError{omdoc}
4451 {The sTeX Global variable #1 is undefined}
4452 {set it with \protect\setSGvar}}
4453 {\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.

```
4454 \*cls)
4455 \@@=mikoslides)
4456 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4457 \RequirePackage{l3keys2e,expl-keystr-compatible}
4458
4459 \keys_define:nn{mikoslides / cls}{
4460   class .code:n = {
4461     \PassOptionsToClass{\CurrentOption}{omdoc}
4462     \str_if_eq:nnT{#1}{book}{
4463       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4464     }
4465     \str_if_eq:nnT{#1}{report}{
4466       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4467     }
4468   },
4469   notes .bool_set:N = \c__mikoslides_notes_bool ,
4470   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4471   unknown .code:n = {
4472     \PassOptionsToClass{\CurrentOption}{omdoc}
4473     \PassOptionsToClass{\CurrentOption}{beamer}
4474     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4475   }
4476 }
4477 \ProcessKeysOptions{ mikoslides / cls }
4478 \bool_if:NTF \c__mikoslides_notes_bool {
4479   \PassOptionsToPackage{notes=true}{mikoslides}
4480 }{
4481   \PassOptionsToPackage{notes=false}{mikoslides}
4482 }
4483 \</cls)
```

now we do the same for the mikoslides package.

```

4484 \package
4485 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4486 \RequirePackage{l3keys2e,expl-keystr-compat}
4487
4488 \keys_define:nn{mikoslides / pkg}{
4489   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4490   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4491   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4492   slides       .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4493   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4494   frameimages  .bool_set:N = \c__mikoslides_frameimages_bool ,
4495   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4496   nopproblems  .bool_set:N = \c__mikoslides_nopproblems_bool,
4497   unknown      .code:n      = {
4498     \PassOptionsToClass{\CurrentOption}{stex}
4499     \PassOptionsToClass{\CurrentOption}{tikzinput}
4500   }
4501 }
4502 \ProcessKeysOptions{ mikoslides / pkg }
4503 \newif\ifnotes
4504 \bool_if:NTF \c__mikoslides_notes_bool {
4505   \notesttrue
4506 }{
4507   \notesfalse
4508 }
4509

```

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

```

4510 \str_if_empty:NTF \c__mikoslides_topsect_str {
4511   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4512 }{
4513   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4514 }
4515 \package

```

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

```

4516 \cls
4517 \bool_if:NTF \c__mikoslides_notes_bool {
4518   \LoadClass{omdoc}
4519 }{
4520   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4521   \newcounter{Item}
4522   \newcounter{paragraph}
4523   \newcounter{subparagraph}
4524   \newcounter{Hfootnote}
4525   \RequirePackage{omdoc}
4526 }

```

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

```

4527 \RequirePackage{mikoslides}
4528 \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.

```

4529 \*package>
4530 \bool_if:NT \c__mikoslides_notes_bool {
4531   \RequirePackage{a4wide}
4532   \RequirePackage{marginnote}
4533   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4534   \RequirePackage{mdframed}
4535   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4536   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4537 }
4538 \RequirePackage{stex-compatibility}
4539 \RequirePackage{stex-tikzinput}
4540 \RequirePackage{etoolbox}
4541 \RequirePackage{amssymb}
4542 \RequirePackage{amsmath}
4543 \RequirePackage{comment}
4544 \RequirePackage{textcomp}
4545 \RequirePackage{url}
4546 \RequirePackage{graphicx}
4547 \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

```

4548 \bool_if:NT \c__mikoslides_notes_bool {
4549   \renewcommand\usetheme[2][\usepackage[#1]{beamernotestheme#2}]
4550 }

```

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

```

4551 \newcounter{slide}
4552 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4553 \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.

```

4554 \bool_if:NTF \c__mikoslides_notes_bool {
4555   \renewenvironment{note}{\ignorespaces}{\ignorespaces}
4556 }{
4557   \excludecomment{note}
4558 }

```

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.

```
4559 \bool_if:NT \c__mikoslides_notes_bool {
4560   \newlength{\slideframewidth}
4561   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4562 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4563   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4564     \bool_set_true:N #1
4565   }{
4566     \bool_set_false:N #1
4567   }
4568 }
4569 \keys_define:nn{mikoslides / frame}{
4570   label .str_set_x:N = \l__mikoslides_frame_label_str,
4571   allowframebreaks .code:n = {
4572     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4573   },
4574   allowdisplaybreaks .code:n = {
4575     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4576   },
4577   fragile .code:n = {
4578     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4579   },
4580   shrink .code:n = {
4581     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4582   },
4583   squeeze .code:n = {
4584     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4585   },
4586   t .code:n = {
4587     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4588   },
4589 }
4590 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4591   \str_clear:N \l__mikoslides_frame_label_str
4592   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4593   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4594   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4595   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4596   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4597   \bool_set_true:N \l__mikoslides_frame_t_bool
4598   \keys_set:nn { mikoslides / frame }{ #1 }
4599 }
```

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

```
4600 \renewenvironment{frame}[1][]{
4601   \__mikoslides_frame_args:n{#1}
4602   \sffamily
4603   \stepcounter{slide}
4604   \def\@currentlabel{\theslide}
4605   \str_if_empty:NF \l__mikoslides_frame_label_str {
4606     \label{\l__mikoslides_frame_label_str}
```

```
4607 }
```

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

```
4608 \def\itemize@level{outer}
4609 \def\itemize@outer{outer}
4610 \def\itemize@inner{inner}
4611 \renewcommand\newpage{\addtocounter{framenum}{1}}
4612 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4613 \renewenvironment{itemize}{
4614   \ifx\itemize@level\itemize@outer
4615     \def\itemize@label{$\rhd$}
4616   \fi
4617   \ifx\itemize@level\itemize@inner
4618     \def\itemize@label{$\scriptstyle\rhd$}
4619   \fi
4620   \begin{list}
4621     {\itemize@label}
4622     {\setlength{\labelsep}{.3em}
4623      \setlength{\labelwidth}{.5em}
4624      \setlength{\leftmargin}{1.5em}
4625     }
4626   \edef\itemize@level{\itemize@inner}
4627 }{
4628   \end{list}
4629 }
```

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

```
4630 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4631 }{
4632   \medskip\miko@slidelabel\end{mdframed}
4633 }
```

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

`\frametitle`

```
4634 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4635 }
```

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

EdN:18

`\pause` 18

```
4636 \bool_if:NT \c__mikoslides_notes_bool {
4637   \newcommand\pause{}
4638 }
```

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

`nomtext`

```
4639 \bool_if:NTF \c__mikoslides_notes_bool {
4640   \newenvironment{nomtext}[1][\begin{sparagraph}[#1]}{\end{sparagraph}}
4641 }{
4642   \excludecomment{nomtext}
4643 }
```

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

```

nomgroup
4644 \bool_if:NTF \c__mikoslides_notes_bool {
4645   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4646 }{
4647   \excludecomment{nomgroup}
4648 }

ndefinition
4649 \bool_if:NTF \c__mikoslides_notes_bool {
4650   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4651 }{
4652   \excludecomment{ndefinition}
4653 }

nassertion
4654 \bool_if:NTF \c__mikoslides_notes_bool {
4655   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4656 }{
4657   \excludecomment{nassertion}
4658 }

nsproof
4659 \bool_if:NTF \c__mikoslides_notes_bool {
4660   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4661 }{
4662   \excludecomment{nsproof}
4663 }

nexample
4664 \bool_if:NTF \c__mikoslides_notes_bool {
4665   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4666 }{
4667   \excludecomment{nexample}
4668 }

\inputref@*skip We customize the hooks for in \inputref.
4669 \def\inputref@preskip{\smallskip}
4670 \def\inputref@postskip{\medskip}

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

\inputref*
4671 \let\orig@inputref\inputref
4672 \def\inputref{\@ifstar\ninputref\orig@inputref}
4673 \newcommand\ninputref[2] [] {
4674   \bool_if:NT \c__mikoslides_notes_bool {
4675     \orig@inputref[#1]{#2}
4676   }
4677 }

(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>}`.

```

4678 \newlength{\slidelogoheight}
4679
4680 \bool_if:NTF \c__mikoslides_notes_bool {
4681   \setlength{\slidelogoheight}{.4cm}
4682 }{
4683   \setlength{\slidelogoheight}{1cm}
4684 }
4685 \newsavebox{\slidelogo}
4686 \sbox{\slidelogo}{\TeX}
4687 \newrobustcmd{\setslidelogo}[1]{
4688   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4689 }
```

(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.

```

4690 \def\source{Michael Kohlhase}% customize locally
4691 \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.

```

4692 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4693 \newsavebox{\cclogo}
4694 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4695 \newif\ifcchref\cchreffalse
4696 \AtBeginDocument{
4697   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4698 }
4699 \def\licensing{
4700   \ifcchref
4701     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4702   \else
4703     {\usebox{\cclogo}}
4704   \fi
4705 }
4706 \newrobustcmd{\setlicensing}[2][]{
4707   \def@url{#1}
4708   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4709   \ifx@url@empty
4710     \def\licensing{{\usebox{\cclogo}}}
4711   \else
4712     \def\licensing{
```

```

4713     \ifcchref
4714     \href{#1}{\usebox{\cclogo}}
4715     \else
4716     {\usebox{\cclogo}}
4717     \fi
4718   }
4719 \fi
4720 }

```

(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

```

4721 \newrobustcmd\miko@slidelabel{
4722   \vbox to \slidelogoheight{
4723     \vss\hbox to \slidewidth
4724     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4725   }
4726 }

```

(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.

```

4727 \def\Gin@mhrepos{}
4728 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4729 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
4730 \newrobustcmd\frameimage[2][{}]{
4731   \stepcounter{slide}
4732   \bool_if:NT \c__mikoslides_frameimages_bool {
4733     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4734     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4735     \begin{center}
4736       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4737         \fbox{
4738           \ifx\Gin@ewidth\@empty
4739             \ifx\Gin@mhrepos\@empty
4740               \mhgraphics[width=\slidewidth,#1]{#2}
4741             \else
4742               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4743             \fi
4744           \else% \Gin@ewidth empty
4745             \ifx\Gin@mhrepos\@empty
4746               \mhgraphics[#1]{#2}
4747             \else
4748               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4749             \fi
4750           \fi% \Gin@ewidth empty
4751         }
4752       }{
4753         \ifx\Gin@ewidth\@empty

```

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


```

4754         \ifx\Gin@mhrepos\empty
4755           \mhgraphics[width=\slidewidth,#1]{#2}
4756         \else
4757           \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4758         \fi
4759         \ifx\Gin@mhrepos\empty
4760           \mhgraphics[#1]{#2}
4761         \else
4762           \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4763         \fi
4764       \fi% Gin@ewidth empty
4765     }
4766     \end{center}
4767     \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4768     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4769   }
4770 } % 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.

```

4771 \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

```

4772 \AddToHook{begindocument}{
4773   \definecolor{green}{rgb}{0,.5,0}
4774   \definecolor{purple}{cmyk}{.3,1,0,.17}
4775 }

```

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`.

```

4776 % \def\STpresent#1{\textcolor{blue}{#1}}
4777 \def\defemph#1{\textcolor{magenta}{#1}}
4778 \def\symrefemph#1{\textcolor{cyan}{#1}}
4779 \def\compemph#1{\textcolor{blue}{#1}}
4780 \def\titleemph#1{\textcolor{blue}{#1}}
4781 \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.

```

4782 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4783 \def\smalltextwarning{
4784   \pgfuseimage{miko@small@dbend}
4785   \xspace
4786 }
4787 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4788 \newrobustcmd\textwarning{
4789   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4790   \xspace
4791 }
4792 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4793 \newrobustcmd\bigtextwarning{
4794   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4795   \xspace
4796 }

```

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

```

4797 \newrobustcmd\putgraphicsat[3]{
4798   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4799 }
4800 \newrobustcmd\putat[2]{
4801   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4802 }

```

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`.

```

4803 \bool_if:NT \c__mikoslides_sectocframes_bool {
4804   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4805     \newcounter{chapter}\counterwithin*{section}{chapter}
4806   }{
4807     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4808       \newcounter{chapter}\counterwithin*{section}{chapter}
4809     }
4810   }
4811 }

```

`\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
4812 \def\part@prefix{}
4813 \@ifpackageloaded{omdoc}{}{
4814   \str_case:VnF \__mikoslidestopsect {
4815     {part}{
4816       \int_set:Nn \l_document_structure_section_level_int {0}
4817       \def\thesection{\arabic{chapter}.\arabic{section}}
4818       \def\part@prefix{\arabic{chapter}.}
4819     }
4820     {chapter}{
4821       \int_set:Nn \l_document_structure_section_level_int {1}
4822       \def\thesection{\arabic{chapter}.\arabic{section}}
4823       \def\part@prefix{\arabic{chapter}.}
4824     }
4825   }{
4826     \int_set:Nn \l_document_structure_section_level_int {2}
4827     \def\part@prefix{}

```

```

4828 }
4829 }
4830
4831 \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

```

4832 \renewenvironment{omgroup}[2][]{
4833   \__document_structure_omgroup_args:n { #1 }
4834   \int_incr:N \l_document_structure_omgroup_level_int
4835   \int_incr:N \l_document_structure_section_level_int
4836   \bool_if:NT \c__mikoslides_sectocframes_bool {
4837     \stepcounter{slide}
4838     \begin{frame}[noframenumbering]
4839     \vfill\Large\centering
4840     \red{
4841       \ifcase\l_document_structure_section_level_int\or
4842         \stepcounter{part}
4843         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4844         \def\currentsectionlevel{\omdoc@part@kw}
4845       \or
4846         \stepcounter{chapter}
4847         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4848         \def\currentsectionlevel{\omdoc@chapter@kw}
4849       \or
4850         \stepcounter{section}
4851         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4852         \def\currentsectionlevel{\omdoc@section@kw}
4853       \or
4854         \stepcounter{subsection}
4855         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4856         \def\currentsectionlevel{\omdoc@subsection@kw}
4857       \or
4858         \stepcounter{subsubsection}
4859         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4860         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4861       \or
4862         \stepcounter{paragraph}
4863         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{paragraph}}
4864         \def\currentsectionlevel{\omdoc@paragraph@kw}
4865       \else
4866         \def\__mikoslideslabel{}
4867         \def\currentsectionlevel{\omdoc@paragraph@kw}
4868       \fi% end ifcase
4869       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4870       \quad #2%
4871     }%
4872     \vfill%
4873     \end{frame}%
4874   }
4875   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%

```

```

4876 }{}
4877 }

```

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

```

4878 \def\inserttheorembodyfont{\normalfont}
4879 \bool_if:NF \c__mikoslides_notes_bool {
4880   \defbeamertemplate{theorem begin}{miko}
4881   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4882     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4883     \inserttheorempunctuation\inserttheorembodyfont\xspace}
4884   \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

4885   \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.

```

4886   \expandafter\def\csname Parent2\endcsname{}
4887 }
4888 \bool_if:NT \c__mikoslides_notes_bool {
4889   \renewenvironment{columns}[1][]{%
4890     \par\noindent%
4891     \begin{minipage}%
4892       \slidewidth\centering\leavevmode%
4893   }{%
4894     \end{minipage}\par\noindent%
4895   }%
4896   \newsavebox\columnbox%
4897   \renewenvironment<>{column}[2][]{%
4898     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4899   }{%
4900     \end{minipage}\end{lrbox}\usebox\columnbox%
4901   }%
4902 }
4903 \bool_if:NTF \c__mikoslides_noproblems_bool {
4904   \newenvironment{problems}{}{}
4905 }{
4906   \excludecomment{problems}
4907 }

```

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.

```

4908 \gdef\printexcursions{}
4909 \newcommand\excursionref[2]{% label, text
4910   \bool_if:NT \c__mikoslides_notes_bool {
4911     \begin{sparagraph}[title=Excursion]
4912       #2 \sref[fallback=the appendix]{#1}.
4913     \end{sparagraph}
4914   }

```

```

4915 }
4916 \newcommand\activate@excursion[2][]{
4917   \gappto\printexcursions{\inputref{#1}{#2}}
4918 }
4919 \newcommand\excursion[4][]{% repos, label, path, text
4920   \bool_if:NT \c__mikoslides_notes_bool {
4921     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4922   }
4923 }

```

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

\excursiongroup

```

4924 \keys_define:nn{mikoslides / excursiongroup }{
4925   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4926   intro       .tl_set:N    = \l__mikoslides_excursion_intro_tl,
4927   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4928 }
4929 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4930   \tl_clear:N \l__mikoslides_excursion_intro_tl
4931   \str_clear:N \l__mikoslides_excursion_id_str
4932   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4933   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4934 }
4935 \newcommand\excursiongroup[1][]{
4936   \__mikoslides_excursion_args:n{ #1 }
4937   \ifdefempty\printexcursions{}% only if there are excursions
4938   {\begin{note}
4939     \begin{omgroup}[#1]{Excursions}%
4940     \ifdefempty\l__mikoslides_excursion_intro_tl{\{
4941       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4942         \l__mikoslides_excursion_intro_tl
4943       }
4944     }
4945     \printexcursions%
4946     \end{omgroup}
4947   \end{note}}
4948 }
4949 \ifcsname beameritemnestingprefix\endcsname\else\def\beameritemnestingprefix{\fi
4950 \</package>

```

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

Chapter 33

The Implementation

33.1 Package Options

The first step is to declare (a few) package options that handle whether certain information is printed or not. They all come with their own conditionals that are set by the options.

```
4951 <*package>
4952 <@@=problems>
4953 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4954 \RequirePackage{l3keys2e,expl-keystr-compat}
4955
4956 \keys_define:nn { problem / pkg }{
4957   notes      .default:n    = { true },
4958   notes      .bool_set:N   = \c__problems_notes_bool,
4959   gnotes     .default:n    = { true },
4960   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4961   hints      .default:n    = { true },
4962   hints      .bool_set:N   = \c__problems_hints_bool,
4963   solutions  .default:n    = { true },
4964   solutions  .bool_set:N   = \c__problems_solutions_bool,
4965   pts        .default:n    = { true },
4966   pts        .bool_set:N   = \c__problems_pts_bool,
4967   min        .default:n    = { true },
4968   min        .bool_set:N   = \c__problems_min_bool,
4969   boxed      .default:n    = { true },
4970   boxed      .bool_set:N   = \c__problems_boxed_bool,
4971   unknown    .code:n       = {}
4972 }
4973 \def\solutionstrue{
4974   \bool_set_true:N \c__problems_solutions_bool
4975 }
4976 \def\solutionsfalse{
4977   \bool_set_false:N \c__problems_solutions_bool
4978 }
4979
4980 \ProcessKeysOptions{ problem / pkg }
```

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

```

4981 \RequirePackage{stex-compatibility}
4982 \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.

```

4983 \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.

```

4984 \def\prob@problem@kw{Problem}
4985 \def\prob@solution@kw{Solution}
4986 \def\prob@hint@kw{Hint}
4987 \def\prob@note@kw{Note}
4988 \def\prob@gnote@kw{Grading}
4989 \def\prob@pt@kw{pt}
4990 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4991 \@ifpackageloaded{babel}{
4992   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4993   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4994     \input{problem-ngerman.ldf}
4995   }
4996   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4997     \input{problem-finnish.ldf}
4998   }
4999   \clist_if_in:NnT \l_tmpa_clist {french}{
5000     \input{problem-french.ldf}
5001   }
5002   \clist_if_in:NnT \l_tmpa_clist {russian}{
5003     \input{problem-russian.ldf}
5004   }
5005 }{}

```

33.2 Problems and Solutions

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

```

5006 \keys_define:nn{ problem / problem }{
5007   id      .str_set:x:N = \l__problems_prob_id_str,
5008   pts     .tl_set:N    = \l__problems_prob_pts_tl,
5009   min     .tl_set:N    = \l__problems_prob_min_tl,
5010   title   .tl_set:N    = \l__problems_prob_title_tl,
5011   refnum  .int_set:N   = \l__problems_prob_refnum_int
5012 }
5013 \cs_new_protected:Nn \__problems_prob_args:n {
5014   \str_clear:N \l__problems_prob_id_str
5015   \tl_clear:N \l__problems_prob_pts_tl
5016   \tl_clear:N \l__problems_prob_min_tl
5017   \tl_clear:N \l__problems_prob_title_tl

```

```

5018 \int_zero_new:N \l__problems_prob_refnum_int
5019 \keys_set:nn { problem / problem }{ #1 }
5020 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
5021   \let\l__problems_inclprob_refnum_int\undefined
5022 }
5023 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

5024 \newcounter{problem}
5025 \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.

```

5026 \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

```

5027 \newcommand\prob@number{
5028   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
5029     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
5030   }{
5031     \int_if_exist:NTF \l__problems_prob_refnum_int {
5032       \prob@label{\int_use:N \l__problems_prob_refnum_int }
5033     }{
5034       \prob@label\theproblem
5035     }
5036   }
5037 }

```

(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.

```

5038 \newcommand\prob@title[3]{%
5039   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
5040     #2 \l__problems_inclprob_title_tl #3
5041   }{
5042     \tl_if_exist:NTF \l__problems_prob_title_tl {
5043       #2 \l__problems_prob_title_tl #3
5044     }{
5045       #1
5046     }
5047   }
5048 }

```

(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.

```

5049 \def\prob@heading{
5050   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
5051   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
5052 }

```

(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`

```

5053 \newenvironment{problem}[1][1]{
5054   \__problems_prob_args:n{#1}%\sref@target%
5055   \@in@omtexttrue% we are in a statement (for inline definitions)
5056   \stepcounter{problem}\record@problem
5057   \def\current@section@level{\prob@problem@kw}
5058   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
5059 }%
5060 {\smallskip}
5061 \bool_if:NT \c__problems_boxed_bool {
5062   \surroundwithmdframed{problem}
5063 }

```

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

```

5064 \def\record@problem{
5065   \protected@write\@auxout{}
5066   {
5067     \string\@problem{\prob@number}
5068     {
5069       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5070         \l__problems_inclprob_pts_tl
5071       }{
5072         \l__problems_prob_pts_tl
5073       }
5074     }%
5075     {
5076       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5077         \l__problems_inclprob_min_tl
5078       }{
5079         \l__problems_prob_min_tl
5080       }
5081     }
5082   }
5083 }

```

(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).

```

5084 \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.

```

5085 \keys_define:nn { problem / solution }{
5086   id          .str_set_x:N = \l__problems_solution_id_str ,
5087   for         .tl_set:N    = \l__problems_solution_for_tl ,
5088   height      .dim_set:N   = \l__problems_solution_height_dim ,
5089   creators    .clist_set:N = \l__problems_solution_creators_clist ,
5090   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
5091   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
5092 }
5093 \cs_new_protected:Nn \__problems_solution_args:n {
5094   \str_clear:N \l__problems_solution_id_str
5095   \tl_clear:N \l__problems_solution_for_tl
5096   \tl_clear:N \l__problems_solution_srccite_tl
5097   \clist_clear:N \l__problems_solution_creators_clist
5098   \clist_clear:N \l__problems_solution_contributors_clist
5099   \dim_zero:N \l__problems_solution_height_dim
5100   \keys_set:nn { problem / solution }{ #1 }
5101 }

```

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

```

5102 \newcommand\@startsolution[1][ ]{
5103   \__problems_solution_args:n { #1 }
5104   \@in@omtexttrue% we are in a statement.
5105   \bool_if:NF \c__problems_boxed_bool { \hrule }
5106   \smallskip\noindent
5107   {\textbf\prob@solution@kw : \enspace}
5108   \begin{small}
5109   \def\current@section@level{\prob@solution@kw}
5110   \ignorespacesandpars
5111 }

```

\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.

```

5112 \newcommand\startsolutions{
5113   \specialcomment{solution}{\@startsolution}{
5114     \bool_if:NF \c__problems_boxed_bool {
5115       \hrule\medskip
5116     }
5117     \end{small}%
5118   }
5119   \bool_if:NT \c__problems_boxed_bool {
5120     \surroundwithmdframed{solution}
5121   }
5122 }

```

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

\stopsolutions

```

5123 \newcommand\stopsolutions{\excludacomment{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.

```

5124 \bool_if:NTF \c__problems_solutions_bool {
5125   \startsolutions
5126 }{
5127   \stopsolutions
5128 }

```

exnote

```

5129 \bool_if:NTF \c__problems_notes_bool {
5130   \newenvironment{exnote}[1][]{
5131     \par\smallskip\hrule\smallskip
5132     \noindent\textbf{\prob@note@kw : }\small
5133   }{
5134     \smallskip\hrule
5135   }
5136 }{
5137   \excludecomment{exnote}
5138 }

```

hint

```

5139 \bool_if:NTF \c__problems_notes_bool {
5140   \newenvironment{hint}[1][]{
5141     \par\smallskip\hrule\smallskip
5142     \noindent\textbf{\prob@hint@kw :~ }\small
5143   }{
5144     \smallskip\hrule
5145   }
5146   \newenvironment{exhint}[1][]{
5147     \par\smallskip\hrule\smallskip
5148     \noindent\textbf{\prob@hint@kw :~ }\small
5149   }{
5150     \smallskip\hrule
5151   }
5152 }{
5153   \excludecomment{hint}
5154   \excludecomment{exhint}
5155 }

```

gnote

```

5156 \bool_if:NTF \c__problems_notes_bool {
5157   \newenvironment{gnote}[1][]{
5158     \par\smallskip\hrule\smallskip
5159     \noindent\textbf{\prob@gnote@kw : }\small
5160   }{
5161     \smallskip\hrule
5162   }
5163 }{
5164   \excludecomment{gnote}
5165 }

```

33.3 Multiple Choice Blocks

```

5166 \newenvironment{mcb}{
5167   \begin{enumerate}
5168 }{
5169   \end{enumerate}
5170 }

```

we define the keys for the mcc macro

```

5171 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
5172   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
5173     \bool_set_true:N #1
5174   }{
5175     \bool_set_false:N #1
5176   }
5177 }
5178 \keys_define:nn { problem / mcc }{
5179   id          .str_set_x:N = \l__problems_mcc_id_str ,
5180   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
5181   T           .default:n   = { true } ,
5182   T           .bool_set:N   = \l__problems_mcc_t_bool ,
5183   F           .default:n   = { true } ,
5184   F           .bool_set:N   = \l__problems_mcc_f_bool ,
5185   Ttext       .code:n       = {
5186     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
5187   } ,
5188   Ftext       .code:n       = {
5189     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
5190   }
5191 }
5192 \cs_new_protected:Nn \l__problems_mcc_args:n {
5193   \str_clear:N \l__problems_mcc_id_str
5194   \tl_clear:N \l__problems_mcc_feedback_tl
5195   \bool_set_true:N \l__problems_mcc_t_bool
5196   \bool_set_true:N \l__problems_mcc_f_bool
5197   \bool_set_true:N \l__problems_mcc_Ttext_bool
5198   \bool_set_false:N \l__problems_mcc_Ftext_bool
5199   \keys_set:nn { problem / mcc }{ #1 }
5200 }

```

\mcc

```

5201 \newcommand\mcc[2][] {
5202   \l__problems_mcc_args:n{ #1 }
5203   \item #2
5204   \bool_if:NT \c__problems_solutions_bool {
5205     \\\
5206     \bool_if:NT \l__problems_mcc_t_bool {
5207       % TODO!
5208       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
5209     }
5210     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

5211      % TODO!
5212      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
5213    }
5214    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
5215      !
5216    }{
5217      \l__problems_mcc_feedback_tl
5218    }
5219  }
5220 } %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.

```

5221
5222 \keys_define:nn{ problem / inclproblem }{
5223   % id      .str_set_x:N = \l__problems_inclprob_id_str,
5224   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
5225   min      .tl_set:N    = \l__problems_inclprob_min_tl,
5226   title    .tl_set:N    = \l__problems_inclprob_title_tl,
5227   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
5228   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
5229 }
5230 \cs_new_protected:Nn \l__problems_inclprob_args:n {
5231   % \str_clear:N \l__problems_prob_id_str
5232   \tl_clear:N \l__problems_inclprob_pts_tl
5233   \tl_clear:N \l__problems_inclprob_min_tl
5234   \tl_clear:N \l__problems_inclprob_title_tl
5235   \int_zero_new:N \l__problems_inclprob_refnum_int
5236   \str_clear:N \l__problems_inclprob_mhrepos_str
5237   \keys_set:nn { problem / inclproblem }{ #1 }
5238   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
5239     \let\l__problems_inclprob_pts_tl\undefined
5240   }
5241   \tl_if_empty:NT \l__problems_inclprob_min_tl {
5242     \let\l__problems_inclprob_min_tl\undefined
5243   }
5244   \tl_if_empty:NT \l__problems_inclprob_title_tl {
5245     \let\l__problems_inclprob_title_tl\undefined
5246   }
5247   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
5248     \let\l__problems_inclprob_refnum_int\undefined
5249   }
5250 }
5251
5252 \cs_new_protected:Nn \l__problems_inclprob_clear: {
5253   % \str_clear:N \l__problems_prob_id_str
5254   \let\l__problems_inclprob_pts_tl\undefined
5255   \let\l__problems_inclprob_min_tl\undefined

```

```

5256 \let\l__problems_inclprob_title_tl\undefined
5257 \let\l__problems_inclprob_refnum_int\undefined
5258 \let\l__problems_inclprob_mhrepos_str\undefined
5259 }
5260
5261 \newcommand\includeproblem[2][\]{
5262   \__problems_inclprob_args:n{ #1 }
5263   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
5264     \input{#2}
5265   }{
5266     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
5267       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
5268     }
5269   }
5270   \__problems_inclprob_clear:
5271 }

```

(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.

```

5272 \AddToHook{enddocument}{
5273   \bool_if:NT \c__problems_pts_bool {
5274     \message{Total:~\arabic{pts}~points}
5275   }
5276   \bool_if:NT \c__problems_min_bool {
5277     \message{Total:~\arabic{min}~minutes}
5278   }
5279 }

```

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

```

5280 \def\pts#1{
5281   \bool_if:NT \c__problems_pts_bool {
5282     \marginpar{#1~\prob@pt@kw}
5283   }
5284 }
5285 \def\min#1{
5286   \bool_if:NT \c__problems_min_bool {
5287     \marginpar{#1~\prob@min@kw}
5288   }
5289 }

```

\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.

```

5290 \newcounter{pts}
5291 \def\show@pts{
5292   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
5293     \bool_if:NT \c__problems_pts_bool {
5294       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
5295       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

5296     }
5297   }{
5298     \tl_if_exist:NT \l__problems_prob_pts_tl {
5299       \bool_if:NT \c__problems_pts_bool {
5300         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
5301         \addtocounter{pts}{\l__problems_prob_pts_tl}
5302       }
5303     }
5304   }
5305 }

```

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

\show@min

```

5306 \newcounter{min}
5307 \def\show@min{
5308   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
5309     \bool_if:NT \c__problems_min_bool {
5310       \marginpar{\l__problems_inclprob_pts_tl;min}
5311       \addtocounter{min}{\l__problems_inclprob_min_tl}
5312     }
5313   }{
5314     \tl_if_exist:NT \l__problems_prob_min_tl {
5315       \bool_if:NT \c__problems_min_bool {
5316         \marginpar{\l__problems_prob_min_tl;min}
5317         \addtocounter{min}{\l__problems_prob_min_tl}
5318       }
5319     }
5320   }
5321 }
5322 \</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.

```
5323 <@@=hwexam>
5324 <*cls>
5325 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5326 \RequirePackage{l3keys2e,expl-keystr-compatible}
5327 \DeclareOption*{
5328   \PassOptionsToClass{\CurrentOption}{omdoc}
5329   \PassOptionsToPackage{\CurrentOption}{stex}
5330   \PassOptionsToPackage{\CurrentOption}{hwexam}
5331   \PassOptionsToPackage{\CurrentOption}{tikzinput}
5332 }
5333 \ProcessOptions
```

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

```
5334 \LoadClass{omdoc}
5335 \RequirePackage{stex}
5336 \RequirePackage{hwexam}
5337 \RequirePackage{tikzinput}
5338 \RequirePackage{graphicx}
5339 \RequirePackage{a4wide}
5340 \RequirePackage{amssymb}
5341 \RequirePackage{amstext}
5342 \RequirePackage{amsmath}
```

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


```

5343 \newcommand\assig@default@type{\hwexam@assignment@kw}
5344 \def\document@hwexamtype{\assig@default@type}
5345 <@@=document_structure>
5346 \keys_define:nn { document-structure / document }{
5347 id .str_set_x:N = \c_document_structure_document_id_str,
5348 hwexamtype .tl_set:N = \document@hwexamtype
5349 }
5350 <@@=hwexam>
5351 </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.

```
5352 \*package>
5353 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
5354 \RequirePackage{l3keys2e,expl-keystr-compat}
5355
5356 \newif\iftest\testfalse
5357 \DeclareOption{test}{\testtrue}
5358 \newif\ifmultiple\multiplefalse
5359 \DeclareOption{multiple}{\multipletrue}
5360 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
5361 \ProcessOptions
```

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

```
5362 \RequirePackage{keyval}[1997/11/10]
5363 \RequirePackage{problem}
```

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

```
5364 \newcommand\hwexam@assignment@kw{Assignment}
5365 \newcommand\hwexam@given@kw{Given}
5366 \newcommand\hwexam@due@kw{Due}
5367 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
5368 space}%
5369 \newcommand\correction@probs@kw{prob.}%
5370 \newcommand\correction@pts@kw{total}%
5371 \newcommand\correction@reached@kw{reached}%
5372 \newcommand\correction@sum@kw{Sum}%
5373 \newcommand\correction@grade@kw{grade}%
5374 \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

```

5375 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
5376
5377 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
5378 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
5379   \input{hwexam-ngerman.ldf}
5380 }
5381 \clist_if_in:NnT \l_tmpa_clist {finnish}{
5382   \input{hwexam-finnish.ldf}
5383 }
5384 \clist_if_in:NnT \l_tmpa_clist {french}{
5385   \input{hwexam-french.ldf}
5386 }
5387 \clist_if_in:NnT \l_tmpa_clist {russian}{
5388   \input{hwexam-russian.ldf}
5389 }

```

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.

```

5390 \newcounter{assignment}
5391 \numberproblemsin{assignment}
5392 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

5393 \keys_define:nn { hwexam / assignment } {
5394   id .str_set:N = \l__hwexam_assign_id_str,
5395   number .int_set:N = \l__hwexam_assign_number_int,
5396   title .tl_set:N = \l__hwexam_assign_title_tl,
5397   type .tl_set:N = \l__hwexam_assign_type_tl,
5398   given .tl_set:N = \l__hwexam_assign_given_tl,
5399   due .tl_set:N = \l__hwexam_assign_due_tl,
5400   loadmodules .code:n = {
5401     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
5402   }
5403 }
5404 \cs_new_protected:Nn \__hwexam_assignment_args:n {
5405   \str_clear:N \l__hwexam_assign_id_str
5406   \int_set:Nn \l__hwexam_assign_number_int {-1}
5407   \tl_clear:N \l__hwexam_assign_title_tl
5408   \tl_clear:N \l__hwexam_assign_type_tl
5409   \tl_clear:N \l__hwexam_assign_given_tl
5410   \tl_clear:N \l__hwexam_assign_due_tl
5411   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5412   \keys_set:nn { hwexam / assignment }{ #1 }
5413 }

```

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.

```

5414 \newcommand\given@due[2]{
5415 \bool_lazy_all:nF {
5416 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5417 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5418 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5419 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5420 }{ #1 }
5421
5422 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5423 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5424 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5425 }
5426 }{
5427 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5428 }
5429
5430 \bool_lazy_or:nnF {
5431 \bool_lazy_and_p:nn {
5432 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5433 }{
5434 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5435 }
5436 }{
5437 \bool_lazy_and_p:nn {
5438 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5439 }{
5440 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5441 }
5442 }{ ,~ }
5443
5444 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5445 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5446 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5447 }
5448 }{
5449 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5450 }
5451
5452 \bool_lazy_all:nF {
5453 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5454 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5455 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5456 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5457 }{ #2 }
5458 }

```

`\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.

```

5459 \newcommand\assignment@title[3]{

```

```

5460 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5461 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5462 #1
5463 }{
5464 #2\l__hwexam_assign_title_tl#3
5465 }
5466 }{
5467 #2\l__hwexam_inclassassign_title_tl#3
5468 }
5469 }

```

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

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

```

5470 \newcommand\assignment@number{
5471 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5472 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5473 \int_use:N \l__hwexam_assign_number_int
5474 }
5475 }{
5476 \int_use:N \l__hwexam_inclassassign_number_int
5477 }
5478 }

```

(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.

```

5479 \newenvironment{assignment}[1][ ]{
5480 \__hwexam_assignment_args:n { #1 }
5481 %\sref@target
5482 \let\__hwexamnum\l__hwexam_assign_number_int
5483 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5484 \stepcounter{assignment}
5485 }{
5486 \setcounter{assignment}{\int_use:N\__hwexamnum}
5487 }
5488 \setcounter{problem}{0}
5489 \def\current@section@level{\document@hwexamtype}
5490 %\sref@label@id{\document@hwexamtype \thesection}
5491 \begin{@assignment}
5492 }{
5493 \end{@assignment}
5494 }

```

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

```

5495 \def\__hwexasstitle{
5496 \protect\document@hwexamtype~\arabic{assignment}
5497 \assignment@title{}\;{} \; -- \given@due{}\}
5498 }

```

```

5499 \ifmultiple
5500 \newenvironment{@assignment}{
5501 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5502 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5503 }{
5504 \begin{omgroup}{\__hwexasstitle}
5505 }
5506 }{
5507 \end{omgroup}
5508 }

```

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

```

5509 \else
5510 \newenvironment{@assignment}{
5511 \begin{center}\bf
5512 \Large\@title\strut\
5513 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5514 \large\given@due{--\;}{\;}{--}
5515 \end{center}
5516 }{}
5517 \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.

```

5518 \keys_define:nn { hwexam / inclassignment } {
5519 %id .str_set_x:N = \l__hwexam_assign_id_str,
5520 number .int_set:N = \l__hwexam_inclassign_number_int,
5521 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5522 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5523 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5524 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5525 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5526 }
5527 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5528 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5529 \tl_clear:N \l__hwexam_inclassign_title_tl
5530 \tl_clear:N \l__hwexam_inclassign_type_tl
5531 \tl_clear:N \l__hwexam_inclassign_given_tl
5532 \tl_clear:N \l__hwexam_inclassign_due_tl
5533 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5534 \keys_set:nn { hwexam / inclassignment }{ #1 }
5535 }
5536 \__hwexam_inclassignment_args:n {}
5537
5538 \newcommand\inputassignment[2][ ]{
5539 \__hwexam_inclassignment_args:n { #1 }
5540 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5541 \input{#2}
5542 }{
5543 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5544 \input{\mhp{path}\l__hwexam_inclasssign_mhrepos_str}{#2}}
5545 }
5546 }
5547 \__hwexam_inclasssign_args:n {}
5548 }
5549 \newcommand\includeassignment[2][ ]{
5550 \newpage
5551 \inputassignment[#1]{#2}
5552 }

```

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

35.4 Typesetting Exams

\quizheading

```

5553 \ExplSyntaxOff
5554 \newcommand\quizheading[1]{%
5555 \def\@tas{#1}%
5556 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5557 \ifx\@tas\empty\else%
5558 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5559 \fi%
5560 }
5561 \ExplSyntaxOn

```

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

\testheading

```

5562 \keys_define:nn { hwexam / testheading } {
5563 min .tl_set:N = \l__hwexam_testheading_min_tl,
5564 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5565 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5566 }
5567 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5568 \tl_clear:N \l__hwexam_testheading_min_tl
5569 \tl_clear:N \l__hwexam_testheading_duration_tl
5570 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5571 \keys_set:nn { hwexam / testheading }{ #1 }
5572 }
5573 \newenvironment{testheading}[1][ ]{
5574 \__hwexam_testheading_args:n{ #1 }
5575 \noindent\large{Name:~\hfill
5576 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5577 \begin{center}
5578 \Large\textbf{\@title}\[1ex]
5579 \large\@date\[3ex]
5580 \end{center}
5581 \textbf{You~have~
5582 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5583 \l__hwexam_testheading_min_tl~minutes
5584 }{
5585 \l__hwexam_testheading_duration_tl
5586 }~

```

```

5587 (sharp)~for~the~test
5588 };\
5589 Write~the~solutions~to~the~sheet.
5590 \par\noindent
5591 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5592 \advance\check@time by -\theassignment@totalmin
5593 The~estimated~time~for~solving~this~exam~is~
5594 {\theassignment@totalmin}~minutes,~
5595 leaving~you~{\the\check@time}~minutes~for~revising~
5596 your~exam.
5597
5598 \par\noindent
5599 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5600 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5601 You~can~reach~{\theassignment@totalpts}~points~if~you~
5602 solve~all~problems.~You~will~only~need~
5603 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5604 i.e.\ {\the\bonus@pts}~points~are~bonus~points.
5605 \vfill
5606 \begin{center}
5607 {
5608 \Large\em You~have~ample~time,~so~take~it~slow~
5609 and~avoid~rushing~to~mistakes!\}[2ex]
5610 Different~problems~test~different~skills~and~
5611 knowledge,~so~do~not~get~stuck~on~one~problem.
5612 }
5613 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5614 \end{center}
5615 }{
5616 \newpage
5617 }

```

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

\testspace

```

5618 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5619 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5620 \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.

```

5621 <@=problems>
5622 \renewcommand\@problem[3]{
5623 \stepcounter{assignment@probs}
5624 \def\__problemspts{#2}

```



```

5625 \ifx\__problemspts\@empty\else
5626 \addtocounter{assignment@totalpts}{#2}
5627 \fi
5628 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5629 \xdef\correction@probs{\correction@probs & #1}%
5630 \xdef\correction@pts{\correction@pts & #2}
5631 \xdef\correction@reached{\correction@reached & }
5632 }
5633 \@@=hwexam>

```

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

\correction@table This macro generates the correction table

```

5634 \newcounter{assignment@probs}
5635 \newcounter{assignment@totalpts}
5636 \newcounter{assignment@totalmin}
5637 \def\correction@probs{\correction@probs@kw}%
5638 \def\correction@pts{\correction@pts@kw}%
5639 \def\correction@reached{\correction@reached@kw}%
5640 \def\after@correction@table{}%
5641 \stepcounter{assignment@probs}
5642 \newcommand\correction@table{
5643 \resizebox{\textwidth}{!}{%
5644 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5645 &\multicolumn{\theassignment@probs}{c|}|%|
5646 {\footnotesize\correction@forgrading@kw} &\\ \hline
5647 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5648 \correction@pts & \theassignment@totalpts & \\ \hline
5649 \correction@reached & & \[.7cm]\hline
5650 \end{tabular}}
5651 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5652 \end{package}

```

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

35.5 Leftovers

at some point, we may want to reactivate the logos font, then we use

here we define the logos that characterize the assignment

```

\font\wierfont=../assignments/wierglas
\font\denkerfont=../assignments/denker
\font\uhrfont=../assignments/uhr
\font\warnschildfont=../assignments/achtung

```

```

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

```