

The \TeX 3 Package *

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

2021-12-25

Abstract

TODO

*Version 3.0 (last revised 2021-12-25)

Contents

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

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

16	problem.sty: An Infrastructure for formatting Problems	51
16.1	Introduction	51
16.2	The User Interface	51
16.2.1	Package Options	51
16.2.2	Problems and Solutions	52
16.2.3	Multiple Choice Blocks	53
16.2.4	Including Problems	53
16.2.5	Reporting Metadata	53
16.3	Limitations	53
17	hwexam.sty/cls: An Infrastructure for formatting Assignments and Exams	55
17.1	Introduction	56
17.2	The User Interface	56
17.2.1	Package and Class Options	56
17.2.2	Assignments	56
17.2.3	Typesetting Exams	56
17.2.4	Including Assignments	57
17.3	Limitations	57
IV	Implementation	59
18	gTeX-Basics Implementation	60
18.1	The gTeXDocument Class	60
18.2	Preliminaries	60
18.3	Messages and logging	61
18.4	Persistence	62
18.5	HTML Annotations	62
18.6	Languages	65
18.7	Activating/Deactivating Macros	66
19	gTeX-MathHub Implementation	67
19.1	Generic Path Handling	67
19.2	PWD and kpsewhich	69
19.3	File Hooks and Tracking	70
19.4	MathHub Repositories	71
20	gTeX-References Implementation	77
20.1	Document URIs and URLs	77
20.2	Setting Reference Targets	79
20.3	Using References	80
21	gTeX-Modules Implementation	81
21.1	The module environment	84
21.2	Invoking modules	89
22	gTeX-Module Inheritance Implementation	91
22.1	SMS Mode	91
22.2	Inheritance	95

23	STEX-Symbols Implementation	100
23.1	Symbol Declarations	100
23.2	Notations	106
24	STEX-Terms Implementation	114
24.1	Symbol Invocations	114
24.2	Terms	116
24.3	Notation Components	123
25	STEX-Structural Features Implementation	126
25.1	The feature environment	126
25.2	Features	128
26	STEX-Statements Implementation	133
26.1	Definitions	134
26.2	Assertions	135
26.3	Examples	137
26.4	OMText	138
27	The Implementation	140
27.1	Package Options	140
27.2	Proofs	140
27.3	Justifications	146
28	STEX-Others Implementation	148
29	STEX-Metatheory Implementation	149
30	Tikzinput Implementation	152
31	document-structure.sty Implementation	154
31.1	The OMDoc Class	154
31.2	Class Options	154
31.3	Beefing up the document environment	155
31.4	Implementation: OMDoc Package	155
31.5	Package Options	155
31.6	Document Structure	157
31.7	Front and Backmatter	160
31.8	Global Variables	162
32	MiKoSlides – Implementation	163
32.1	Class and Package Options	163
32.2	Notes and Slides	165
32.3	Header and Footer Lines	169
32.4	Frame Images	170
32.5	Colors and Highlighting	171
32.6	Sectioning	172
32.7	Excursions	174

33 The Implementation	176
33.1 Package Options	176
33.2 Problems and Solutions	177
33.3 Multiple Choice Blocks	182
33.4 Including Problems	183
33.5 Reporting Metadata	184
34 Implementation: The hwexam Class	186
34.1 Class Options	186
35 Implementation: The hwexam Package	188
35.1 Package Options	188
35.2 Assignments	189
35.3 Including Assignments	192
35.4 Typesetting Exams	193
35.5 Leftovers	195

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX` Both print this \TeX logo.
`\stex`

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

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

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

Example 7

```
\mult![\comp{Multiplication}] (denoted by  $\mathbin{\textcolor{teal}{*}}!\mathbin{\textcolor{teal}{\cdot}}$ ) 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. $\int \int \int f \, dx \, dy \, dz$?

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 S^CA^LL^AT_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>	$\underline{N}\underline{T}\underline{F}$ \star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

.

5.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

.

Test 5

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

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

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

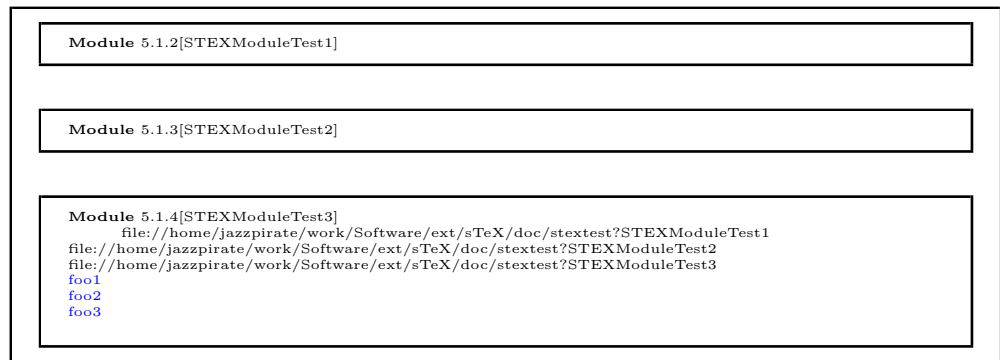
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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

`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

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

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

Test 7

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

6.1.2 Imports and Inheritance

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

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

Test 8

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

Module 6.1.1[Foo]
Meaning: $\text{\macro:-}\text{\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}\text{\<}$

Meaning: $\text{\macro:-}\text{\protect \bar \<}$

Module 6.1.2[Importtest]
Meaning: $\text{\macro:-}\text{\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}\text{\<}$

Module 6.1.3[Importtest2]
Meaning: $\text{\macro:-}\text{\stex_invoke_symbol:n \{file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?Foo?foo\}}\text{\<}$

`\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]
Meaning: `\macro:~>\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest1?foo}<`

Module 6.1.6[UseTest3]
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?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collect`, `http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?aseqfromto`, `http://mathhub.info/sTeX?Metatheory?aseqfromtovia`, `http://mathhub.info/sTeX?Metatheory?module-type`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?UseTest2?bar`

Test 10

```

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

```

Circular dependencies:

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

`\stex_import_module_uri:nn`

`\stex_import_module_uri:nn {⟨archive-ID⟩} {⟨module-path⟩}`

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

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

1. If `⟨archive-ID⟩` is empty:

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

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

2. Otherwise:

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

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

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

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

`\stex_import_require_module:nnnn`

`{⟨ns⟩} {⟨archive-ID⟩} {⟨path⟩} {⟨name⟩}`

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

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

Chapter 7

STEX-Symbols

Code related to symbol declarations and notations

7.1 Macros and Environments

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

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

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

`\stex_symdecl_do:n`

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

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

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

Test 11

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

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

`\l_stex_all_symbols_seq`

Stores full URIs for all modules currently in scope.

`\stex_get_symbol:n`

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

`\notation`

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

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

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

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

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

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

Test 12

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

Module 7.1.2[NotationTest]

`\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/>	<code>\infprec</code> <code>\neginfprec</code>	Maximal and minimal notation precedences.
<hr/> <hr/>	<code>\dobrackets</code>	<code>\dobrackets {⟨body⟩}</code> Puts $\langle body \rangle$ in parentheses; scaled if in display mode unscaled otherwise. Uses the current \S I E X brackets (by default (and)), which can be changed temporarily using <code>\withbrackets</code> .
<hr/> <hr/>	<code>\withbrackets</code>	<code>\withbrackets ⟨left⟩ ⟨right⟩ {⟨body⟩}</code> Temporarily (i.e. within $\langle body \rangle$) sets the brackets used by \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 <code>\left</code> and <code>\right</code> 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]
 $\langle a^b c \rangle$ and $\langle a^b c \rangle$.

Test 15

```

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

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

```

Module 8.1.2[MathTest2]
 $\langle a|[b;c,d,e,f]^g \rangle$ 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]
  some a and some b and also some c here.
  some a and some b and also some c here.

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

`\comp{<args>}`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

mathstructure TODO

Test 17

```

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 9.1.1[StructureTest1]

a**o**b:*M*

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

feature?op

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

Test: a+b

Test2: *(U,+)*

Chapter 10

TeX-Statements

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

10.1 Macros and Environments

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

Chapter 11

sTeX-Proofs: Structural Markup for Proofs

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

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

Contents

11.1 Introduction

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

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

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

Example 1: A very explicit proof, marked up semantically

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

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

11.2 The User Interface

11.2.1 Package Options

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

11.2.2 Proofs and Proof steps

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

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

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

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

11.2.3 Justifications

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

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

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

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

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

11.2.4 Proof Structure

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

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

Chapter 12

sTeX-Metatheory

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

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

12.1 Symbols

Part III
Extensions

Chapter 13

Tikzinput

13.1 Macros and Environments

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

Chapter 14

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

The `omdoc` package is part of the $\S\mathrm{T}_{\mathrm{E}}\mathrm{X}$ collection, a version of $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ that allows to markup $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_{\mathrm{E}}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ into a document format for mathematical knowledge management (MKM).

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

14.1 Introduction

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

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

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

14.2 The User Interface

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

14.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

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

14.2.2 Document Structure

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

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

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

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

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

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

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

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

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

Example 3: A typical Document Structure of a Book

`\skipomgroup`

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

`\currentsectionlevel`

`\CurrentSectionLevel`

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

14.2.3 Ignoring Inputs

`ignore`
`showignores`

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

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

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

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

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

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

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

14.2.4 Structure Sharing

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

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

14.2.5 Global Variables

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

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

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

14.2.6 Colors

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

14.3 Limitations

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

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

Chapter 15

Slides and Course Notes

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

15.1 Introduction

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

15.2 The User Interface

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

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

15.2.1 Package Options

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

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

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

15.2.2 Notes and Slides

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

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

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

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

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

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

Example 4: A typical Course Notes File

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

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

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

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

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

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

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

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

15.2.3 Header and Footer Lines of the Slides

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

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

15.2.4 Frame Images

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

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

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

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

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


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

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

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

`\mhframeimage{baz/foobar}`

15.2.5 Colors and Highlighting

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

15.2.6 Front Matter, Titles, etc.

15.2.7 Excursions

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

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

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

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

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

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

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

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

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

15.2.8 Miscellaneous

15.3 Limitations

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

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

Chapter 16

problem.sty: An Infrastructure for formatting Problems

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

16.1 Introduction

The `problem` package supplies an infrastructure that allows specify problem. Problems are text fragments that come with auxiliary functions: hints, notes, and solutions⁵. Furthermore, we can specify how long the solution to a given problem is estimated to take and how many points will be awarded for a perfect solution.

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

16.2 The User Interface

16.2.1 Package Options

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

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

16.2.2 Problems and Solutions

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

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

Example 5: A marked up Problem

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

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


---


Hint: Think positively, this is simple!


---


Note:Justify your answer


---


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


---


```

Example 6: The Formatted Problem from Figure 5

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

exnote

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

may arise in grading.

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

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

16.2.3 Multiple Choice Blocks

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

<code>T</code>	• <code>T</code> for true answers, <code>F</code> for false ones,
<code>F</code>	
<code>Ttext</code>	• <code>Ttext</code> the verdict for true answers, <code>Ftext</code> for false ones, and
<code>Ftext</code>	
<code>feedback</code>	• <code>feedback</code> for a short feedback text given to the student.

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:

2021-12-25

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

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

[illegible]

Example 8: A generated test heading.

Part IV

Implementation

Chapter 18

STEX -Basics Implementation

18.1 The STEXDocument Class

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

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

18.2 Preliminaries

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

```

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

```

\stex The \TeX logo:

\sTeX

```

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

```

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

18.3 Messages and logging

```

40 <@@=stex_log>

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

```

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

```

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

```

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

Redirecting messages:

```

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

```

18.4 Persistence

75 $\langle @@=stex_persist \rangle$

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

```

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

```

(End definition for $\backslash c_stex_persist_sms_iow$.)

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

```

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

```

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

18.5 HTML Annotations

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

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

```

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

```

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

```

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

```

```

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

```

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

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

```

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

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

```

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

```

(End definition for __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool Whether to (locally) produce HTML output

```

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

```

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

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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

We define four macros for introducing attributes in the HTML output. The definitions depend on the “backend” used (L^AT_EXML, S^CA^LT_EX, p^Df^Lat_Ex).

The p^Df^Lat_Ex-macros largely do nothing; the S^CA^LT_EX-implementations are pretty clear in what they do, the L^AT_EXML-implementations resort to perl bindings.

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

```

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

```

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

18.6 Languages

```

219 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

```

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

```

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

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

```

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

```

18.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

```

266 </package>

```


Chapter 19

STEX -MathHub Implementation

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

19.1 Generic Path Handling

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

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

```

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

```

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

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

```

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

```

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

`\c__stex_path_dot_str`
`\c__stex_path_up_str`

. and .., respectively.

```

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

```

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

`\stex_path_canonicalize:N`

Canonicalizes the path provided; in particular, resolves . and .. path segments.

```

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

```

```

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

```

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

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

```

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

```

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

19.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

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

We determine the PWD

`\c_stex_pwd_seq`
`\c_stex_pwd_str`

```

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

```

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

19.3 File Hooks and Tracking

372 `<@@=stex_files>`

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

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

`\g__stex_files_stack` keeps track of file changes

373 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

375 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

376 `\c_stex_mainfile_str`

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

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

```

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

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

19.4 MathHub Repositories

```

401 <@@=stex_mathhub>

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

```

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

```

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

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
438 \str_new:N\l__stex_mathhub_manifest_file_seq

```

(End definition for `\l__stex_mathhub_manifest_file_seq`.)

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

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

474 \ior_new:N \c_stex_mathhub_manifest_ior

```

(End definition for `\c_stex_mathhub_manifest_ior`.)

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

```

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

```

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n`.)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

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

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

```

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

```

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

\inputref
\inputref:nn

```

572 \newif \ifinputref \inputreffalse
573
574 \cs_new_protected:Nn \inputref:nn {
575   \stex_in_repository:nn {#1} {
576     \ifinputref
577       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
578     \else
579       \inputreftrue
580       \input{ \c_stex_mathhub_str / ##1 / source / #2 }
581     \inputreffalse
582   \fi
583 }
584 }
585 \NewDocumentCommand \inputref { 0{} m}{
586   \inputref:nn{ #1 }{ #2 }
587 }

```

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

\mhp

```

588 \def \mhp #1 #2 {
589   \exp_args:Ne \str_if_eq:nnTF{#1}{}{
590     \c_stex_mathhub_str /
591     \prop_item:Nn \l_stex_current_repository_prop { id }
592     / source / #2
593   }{
594     \c_stex_mathhub_str / #1 / source / #2
595   }
596 }

```

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

\libinput

```

597 \cs_new_protected:Npn \libinput #1 {
598   \prop_get:NnNF \l_stex_current_repository_prop {id} \l_tmpa_str {
599     \msg_error:nnn{stex}{error/notinarchive}\libinput
600   }
601   \bool_set_false:N \l_tmpa_bool
602   \tl_clear:N \l_tmpa_tl
603   \seq_set_eq:NN \l_tmpa_seq \c_stex_mathhub_seq
604   \seq_set_split:NnV \l_tmpb_seq / \l_tmpa_str
605   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str
606   \seq_pop_left:NNT \l_tmpb_seq \l_tmpb_str {
607     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
608     \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
609       / meta-inf / lib / #1.tex}{
610       \bool_set_true:N \l_tmpa_bool
611       \tl_put_right:Nx \l_tmpa_tl {
612         \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
613           / meta-inf / lib / #1.tex}
614       }
615     }{}
616   }

```

```

617 \IfFileExists{ \stex_path_to_string:N \l_tmpa_seq
618 / \l_tmpa_str / lib / #1.tex
619 }{
620   \bool_set_true:N \l_tmpa_bool
621   \tl_put_right:Nx \l_tmpa_tl {
622     \exp_not:N \input { \stex_path_to_string:N \l_tmpa_seq
623       / \l_tmpa_str / lib / #1.tex}
624   }
625 }{}
626 \bool_if:NF \l_tmpa_bool {
627   \msg_error:nnnn{stex}{error/nofile}\libinput{#1.tex}
628 }
629 \l_tmpa_tl
630 }

```

(End definition for `\libinput`. This function is documented on page [13](#).)

```

631 \</package>

```

Chapter 20

STEX -References Implementation

```
632 <*package>
633
634 %%%%%%%%%% references.dtx %%%%%%%%%%
635
636 %\RequirePackage{hyperref}
637 %\RequirePackage{cleveref}
638 <@@=stex_refs>
639
640 \iow_new:N \c__stex_refs_refs_iow
641 \AddToHook{begindocument}{
642   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
643 }
644 \AddToHook{enddocument}{
645   \iow_close:N \c__stex_refs_refs_iow
646 }
647
648 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
649
650 \NewDocumentCommand \STEXreftitle { m } {
651   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
652 }
```

20.1 Document URIs and URLs

```
653 \seq_new:N \g__stex_refs_all_refs_seq
654
655 \str_new:N \l_stex_current_docns_str
656
657 \cs_new_protected:Nn \stex_get_document_uri: {
658   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
659   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
660   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
661   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
```

```

662 \seq_put_right:No \l_tmpa_seq \l_tmpb_str
663
664 \str_clear:N \l_tmpa_str
665 \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
666   \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
667 }
668
669 \str_if_empty:NTF \l_tmpa_str {
670   \str_set:Nx \l_stex_current_docns_str {
671     file:/\stex_path_to_string:N \l_tmpa_seq
672   }
673 }{
674   \bool_set_true:N \l_tmpa_bool
675   \bool_while_do:Nn \l_tmpa_bool {
676     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
677     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
678       {source} { \bool_set_false:N \l_tmpa_bool }
679     }{}{
680       \seq_if_empty:NT \l_tmpa_seq {
681         \bool_set_false:N \l_tmpa_bool
682       }
683     }
684   }
685
686   \seq_if_empty:NTF \l_tmpa_seq {
687     \str_set_eq:NN \l_stex_current_docns_str \l_tmpa_str
688   }{
689     \str_set:Nx \l_stex_current_docns_str {
690       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
691     }
692   }
693 }
694 }
695
696 \str_new:N \l_stex_current_docurl_str
697 \cs_new_protected:Nn \stex_get_document_url: {
698   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
699   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
700   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
701   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
702   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
703
704   \str_clear:N \l_tmpa_str
705   \prop_get:NnNF \l_stex_current_repository_prop { docurl } \l_tmpa_str {
706     \prop_get:NnNF \l_stex_current_repository_prop { narr } \l_tmpa_str {
707       \prop_get:NnNF \l_stex_current_repository_prop { ns } \l_tmpa_str {}
708     }
709   }
710
711   \str_if_empty:NTF \l_tmpa_str {
712     \str_set:Nx \l_stex_current_docurl_str {
713       file:/\stex_path_to_string:N \l_tmpa_seq
714     }
715   }{
716     \bool_set_true:N \l_tmpa_bool

```

```

716 \bool_while_do:Nn \l_tmpa_bool {
717   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
718   \exp_args:No \str_case:nnTF { \l_tmpb_str } {
719     {source} { \bool_set_false:N \l_tmpa_bool }
720   }{}{
721     \seq_if_empty:NT \l_tmpa_seq {
722       \bool_set_false:N \l_tmpa_bool
723     }
724   }
725 }
726
727 \seq_if_empty:NTF \l_tmpa_seq {
728   \str_set_eq:NN \l_stex_current_docurl_str \l_tmpa_str
729 }{
730   \str_set:Nx \l_stex_current_docurl_str {
731     \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
732   }
733 }
734 }
735 }

```

20.2 Setting Reference Targets

```

736 \str_const:Nn \c__stex_refs_url_str{URL}
737 \str_const:Nn \c__stex_refs_ref_str{REF}
738 % @currentlabel -> number
739 % @currentlabelname -> title
740 % @currentHref -> name.number <- id of some kind
741 % \theH# -> \arabic{section}
742 % \the# -> number
743 % \hyper@makecurrent{#}
744 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
745   \stex_get_document_uri:
746   \str_set:Nx \l_tmpa_str { #1 }
747   \str_if_empty:NT \l_tmpa_str {
748     \int_zero:N \l_tmpa_int
749     \bool_set_true:N \l_tmpa_bool
750     \bool_while_do:Nn \l_tmpa_bool {
751       \cs_if_exist:cTF {
752         sref_\l_stex_current_docns_str\c_hash_str REF_\int_use:N \l_tmpa_int _type
753       }{
754         \int_incr:N \l_tmpa_int
755       }{
756         \str_set:Nx \l_tmpa_str { REF_\int_use:N \l_tmpa_int }
757         \bool_set_false:N \l_tmpa_bool
758       }
759     }
760   }
761   \str_set:Nx \l_tmpa_str {
762     \l_stex_current_docns_str\c_hash_str\l_tmpa_str
763   }
764   \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
765   \stex_if_smsmode:TF {
766     \stex_get_document_url:

```

```

767 \str_gset_eq:cN {sref_url_\l_tmpa_str_str}\l_stex_current_docurl_str
768 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_url_str
769 }{
770 \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel~in~\exp_a
771 \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
772 \str_gset_eq:cN {sref_\l_tmpa_str_type}\c__stex_refs_ref_str
773 }
774 }

775 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
776 \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
777 }

```

20.3 Using References

```

778 \keys_define:nn { stex / sref } {
779 linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
780 fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
781 pre           .tl_set:N = \l__stex_refs_pre_tl ,
782 post          .tl_set:N = \l__stex_refs_post_tl ,
783 %indoc        .str_set_x:N = \l__stex_refs_repo_str ,
784 }
785
786 \cs_new_protected:Nn \__stex_refs_args:n {
787 \tl_clear:N \l__stex_refs_linktext_tl
788 \tl_clear:N \l__stex_refs_fallback_tl
789 \tl_clear:N \l__stex_refs_pre_tl
790 \tl_clear:N \l__stex_refs_post_tl
791 \str_clear:N \l__stex_refs_repo_str
792 \keys_set:nn { stex / sref } { #1 }
793 }
794
795 \</package>

```

Chapter 21

STEX -Modules Implementation

```
796 <*package>
797
798 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
799
800 <@@=stex_modules>
801
802 Warnings and error messages
803 \msg_new:nnn{stex}{error/unknownmodule}{
804   No~module~#1~found
805 }
806 \msg_new:nnn{stex}{error/syntax}{
807   Syntax~error:~#1
808 }
809 \msg_new:nnn{stex}{error/siglanguage}{
810   Module~#1~declares~signature~#2,~but~does~not~
811   declare~its~language
812 }
```

`\l_stex_current_module_prop` The current module:

```
811 \prop_new:N \l_stex_current_module_prop
```

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

`\l_stex_all_modules_seq` Stores all available modules

```
812 \seq_new:N \l_stex_all_modules_seq
```

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

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

```
813 \seq_new:N \g_stex_modules_in_file_seq
814 \prop_new:N \g_stex_module_files_prop
```

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

```

\stex_if_in_module_p:
\stex_if_in_module:TF
815 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
816   \prop_if_empty:NTF \l_stex_current_module_prop
817   \prg_return_false: \prg_return_true:
818 }

```

(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
819 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
820   \prop_if_exist:cTF { c_stex_module_#1_prop }
821   \prg_return_true: \prg_return_false:
822 }

```

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

```

\stex_add_to_current_module:n
\STEXexport
823 \cs_new_protected:Nn \stex_add_to_current_module:n {
824   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
825   \tl_put_right:Nn \l_tmpa_tl { #1 }
826   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
827 }
828 \cs_new_protected:Npn \STEXexport {
829   \begingroup
830   \newlinechar=-1\relax
831   \endlinechar=-1\relax
832   %\catcode'\ = 9\relax
833   \expandafter\endgroup\STEXexport:n
834 }
835 \cs_new_protected:Nn \STEXexport:n {
836   \ignorespaces #1
837   \stex_add_to_current_module:n { \ignorespaces #1 }
838   \stex_smsmode_set_codes:
839 }
840 \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
841 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
842   \str_set:Nx \l_tmpa_str { #1 }
843   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
844   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
845   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
846 }

```

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

```

\stex_add_import_to_current_module:n
847 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
848   \str_set:Nx \l_tmpa_str { #1 }
849   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
850   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
851   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
852 }

```


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

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

```

853 \cs_new_protected:Nn \stex_modules_compute_namespace:nN {
854   \str_set:Nx \l_tmpa_str { #1 }
855   \seq_set_eq:NN \l_tmpa_seq #2
856   % split off file extension
857   \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
858   \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
859   \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
860   \seq_put_right:No \l_tmpa_seq \l_tmpb_str
861
862   \bool_set_true:N \l_tmpa_bool
863   \bool_while_do:Nn \l_tmpa_bool {
864     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
865     \exp_args:No \str_case:nnTF { \l_tmpb_str } {
866       {source} { \bool_set_false:N \l_tmpa_bool }
867     }{}{
868       \seq_if_empty:NT \l_tmpa_seq {
869         \bool_set_false:N \l_tmpa_bool
870       }
871     }
872   }
873
874   \seq_if_empty:NTF \l_tmpa_seq {
875     \str_set_eq:NN \l_stex_modules_ns_str \l_tmpa_str
876   }{
877     \str_set:Nx \l_stex_modules_ns_str {
878       \l_tmpa_str/\stex_path_to_string:N \l_tmpa_seq
879     }
880   }
881 }

```

(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`

```

882 \str_new:N \l_stex_modules_ns_str

```

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

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

```

883 \cs_new_protected:Nn \stex_modules_current_namespace: {
884   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
885     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
886   }{
887     % split off file extension
888     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
889     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
890     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
891     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
892     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

893     \str_set:Nx \l_stex_modules_ns_str {
894         file:/\stex_path_to_string:N \l_tmpa_seq
895     }
896 }
897 }

```

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

21.1 The module environment

module arguments:

```

898 \keys_define:nn { stex / module } {
899     title          .str_set_x:N = \l_stex_module_title_str ,
900     ns             .str_set_x:N = \l_stex_module_ns_str ,
901     lang           .str_set_x:N = \l_stex_module_lang_str ,
902     sig            .str_set_x:N = \l_stex_module_sig_str ,
903     creators       .str_set_x:N = \l_stex_module_creators_str ,
904     contributors   .str_set_x:N = \l_stex_module_contributors_str ,
905     meta           .str_set_x:N = \l_stex_module_meta_str
906 }
907
908 \cs_new_protected:Nn \__stex_modules_args:n {
909     \str_clear:N \l_stex_module_title_str
910     \str_clear:N \l_stex_module_ns_str
911     \str_clear:N \l_stex_module_lang_str
912     \str_clear:N \l_stex_module_sig_str
913     \str_clear:N \l_stex_module_creators_str
914     \str_clear:N \l_stex_module_contributors_str
915     \str_clear:N \l_stex_module_meta_str
916     \keys_set:nn { stex / module } { #1 }
917 }
918
919 % module parameters here? In the body?
920

```

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

```

921 \cs_new_protected:Nn \stex_module_setup:nn {
922     \str_set:Nx \l_stex_module_name_str { #2 }
923     \__stex_modules_args:n { #1 }
924
925     First, we set up the name and namespace of the module.
926     Are we in a nested module?
927
928     \stex_if_in_module:TF {
929         % Nested module
930         \prop_get:NnN \l_stex_current_module_prop
931         { ns } \l_stex_module_ns_str
932         \str_set:Nx \l_stex_module_name_str {
933             \prop_item:Nn \l_stex_current_module_prop
934             { name } / \l_stex_module_name_str
935         }
936     }{
937         % not nested:
938         \str_if_empty:NT \l_stex_module_ns_str {

```

```

935     \stex_modules_current_namespace:
936     \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
937     \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
938       / {\l_stex_module_ns_str}
939     \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
940     \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
941       \str_set:Nx \l_stex_module_ns_str {
942         \stex_path_to_string:N \l_tmpa_seq
943       }
944     }
945   }
946 }

```

Next, we determine the language of the module:

```

947 \str_if_empty:NT \l_stex_module_lang_str {
948   \seq_get_right:NN \g_stex_currentfile_seq \l_tmpa_str
949   \seq_set_split:NnV \l_tmpa_seq . \l_tmpa_str
950   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str % .tex
951   \seq_pop_left:NN \l_tmpa_seq \l_tmpa_str % <filename>
952   \seq_if_empty:NF \l_tmpa_seq { %remaining element should be language
953     \stex_debug:nn{modules} {Language~\l_stex_module_lang_str~
954       inferred~from~file~name}
955     \seq_pop_left:NN \l_tmpa_seq \l_stex_module_lang_str
956   }
957 }
958
959 \str_if_empty:NF \l_stex_module_lang_str {
960   \prop_get:NVNTF \c_stex_languages_prop \l_stex_module_lang_str
961   \l_tmpa_str {
962     \ltx@ifpackageloaded{babel}{
963       \exp_args:Nx \selectlanguage { \l_tmpa_str }
964     }{}
965   } {
966     \msg_error:nnn{stex}{error/unknownlanguage}{\l_tmpa_str}
967   }
968 }

```

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

```

969 \str_if_empty:NTF \l_stex_module_sig_str {
970   \str_clear:N \l_tmpa_str
971   \seq_clear:N \l_tmpa_seq
972   \tl_clear:N \l_tmpa_tl
973   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
974     name      = \l_stex_module_name_str ,
975     ns        = \l_stex_module_ns_str ,
976     imports   = \exp_not:o { \l_tmpa_seq } ,
977     constants = \exp_not:o { \l_tmpa_seq } ,
978     content   = \exp_not:o { \l_tmpa_tl } ,
979     file      = \exp_not:o { \g_stex_currentfile_seq } ,
980     lang      = \l_stex_module_lang_str ,
981     sig       = \l_stex_module_sig_str ,
982     meta      = \l_stex_module_meta_str
983   }

```

```

984 }{
985   \str_if_empty:NT \l_stex_module_lang_str {
986     \msg_error:nnnn{stex}{error/siglanguage}{
987       \l_stex_module_ns_str?\l_stex_module_name_str
988     }\l_stex_module_sig_str}
989   }
990
991   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
992   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
993   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
994   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
995   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
996   \str_set:Nx \l_tmpa_str {
997     \stex_path_to_string:N \l_tmpa_seq /
998     \l_tmpa_str . \l_stex_module_sig_str .tex
999   }
1000   \IfFileExists \l_tmpa_str {
1001     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
1002       \seq_clear:N \l_stex_all_modules_seq
1003       \prop_clear:N \l_stex_current_module_prop
1004       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
1005       \input { \l_tmpa_str }
1006     }
1007   }{
1008     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1009   }
1010   \stex_activate_module:n {
1011     \l_stex_module_ns_str ? \l_stex_module_name_str
1012   }
1013   \prop_set_eq:Nc \l_stex_current_module_prop {
1014     c_stex_module_
1015     \l_stex_module_ns_str ?
1016     \l_stex_module_name_str
1017     _prop
1018   }
1019 }

```

We load the metatheory:

```

1020 \str_if_empty:NT \l_stex_module_meta_str {
1021   \str_set:Nx \l_stex_module_meta_str {
1022     \c_stex_metatheory_ns_str ? Metatheory
1023   }
1024 }
1025 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1026   \exp_args:Nx \stex_add_to_current_module:n {
1027     \stex_activate_module:n {\l_stex_module_meta_str}
1028   }
1029   \stex_activate_module:n {\l_stex_module_meta_str}
1030 }
1031 }

```

(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}

1032 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1033   \stex_reactivate_macro:N \STEXexport
1034   \stex_reactivate_macro:N \importmodule
1035   \stex_reactivate_macro:N \symdecl
1036   \stex_reactivate_macro:N \notation
1037   \stex_reactivate_macro:N \symdef
1038   \stex_module_setup:nn{#1}{#2}
1039
1040   \stex_debug:nn{modules}{
1041     New~module:\\
1042     Namespace:~\l_stex_module_ns_str\\
1043     Name:~\l_stex_module_name_str\\
1044     Language:~\l_stex_module_lang_str\\
1045     Signature:~\l_stex_module_sig_str\\
1046     Metatheory:~\l_stex_module_meta_str\\
1047     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1048   }
1049
1050   \seq_put_right:Nx \l_stex_all_modules_seq {
1051     \l_stex_module_ns_str ? \l_stex_module_name_str
1052   }
1053
1054   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1055     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1056
1057   \stex_if_smsmode:TF {
1058     \stex_smsmode_set_codes:
1059   } {
1060     \begin{stex_annotate_env} {theory} {
1061       \l_stex_module_ns_str ? \l_stex_module_name_str
1062     }
1063
1064     \stex_annotate_invisible:nnn{header}{} {
1065       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1066       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1067       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1068         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1069       }
1070     }
1071   }
1072   % TODO: Inherit metatheory for nested modules?
1073 }
1074 \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}

1075 \cs_new_protected:Nn \__stex_modules_end_module: {
1076   \str_set:Nx \l_tmpa_str {
1077     c_stex_module_
1078     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1079     \prop_item:Nn \l_stex_current_module_prop { name }
1080     _prop

```

```

1081 }
1082 %^^A \prop_new:c { \l_tmpa_str }
1083 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1084 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1085 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1086 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1087 \NewDocumentEnvironment { @module } { 0{} m } {
1088   \par
1089   \_stex_modules_begin_module:nn{#1}{#2}
1090 } {
1091   \_stex_modules_end_module:
1092   \stex_if_smsmode:TF {
1093     \exp_args:Nx \stex_add_to_sms:n {
1094       \prop_gset_from_keyval:cn {
1095         c_stex_module_
1096         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1097         \prop_item:Nn \l_stex_current_module_prop { name }
1098         _prop
1099       } {
1100         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1101         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1102         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1103         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1104         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1105         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1106         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1107         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1108         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1109       }
1110     }
1111   }{
1112     \end{stex_annotate_env}
1113   }
1114 }

```

\stex_modules_heading: Code for document headers

```

1115 \cs_if_exist:NTF \thesection {
1116   \newcounter{module}[section]
1117 }{
1118   \newcounter{module}
1119 }
1120
1121 \bool_if:NT \c_stex_showmods_bool {
1122   \latexml_if:F { \RequirePackage{mdframed} }
1123 }
1124
1125 \cs_new_protected:Nn \stex_modules_heading: {
1126   \stepcounter{module}
1127   \par
1128   \bool_if:NT \c_stex_showmods_bool {

```

```

1129     \noindent{\textbf{Module} ~
1130       \cs_if_exist:NT \thesection {\thesection.}
1131       \themodule ~ [\l_stex_module_name_str]
1132     }
1133     \str_if_empty:NTF \l_stex_module_title_str {
1134     }{
1135       \quad(\l_stex_module_title_str)\hfill
1136     }\par
1137   }
1138   \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1139   % TODO
1140   \stex_ref_new_doc_target:n \l_stex_module_name_str
1141 }

```

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

Finally:

```

1142 \NewDocumentEnvironment { module } { 0{} m } {
1143   \bool_if:NT \c_stex_showmods_bool {
1144     \begin{mdframed}
1145   }
1146   \begin{@module}[#1]{#2}
1147   \stex_modules_heading:
1148 }{
1149   \end{@module}
1150   \bool_if:NT \c_stex_showmods_bool {
1151     \end{mdframed}
1152   }
1153 }

```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1154 \NewDocumentCommand \STEXModule { m } {
1155   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1156   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1157   \tl_set:Nn \l_tmpa_tl {
1158     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1159   }
1160   \seq_map_inline:Nn \l_stex_all_modules_seq {
1161     \str_set:Nn \l_tmpb_str { ##1 }
1162     \str_if_eq:eeT { \l_tmpa_str } {
1163       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1164     } {
1165       \seq_map_break:n {
1166         \tl_set:Nn \l_tmpa_tl {
1167           \stex_invoke_module:n { ##1 }
1168         }
1169       }
1170     }
1171   }
1172   \l_tmpa_tl
1173 }
1174

```

```

1175 \cs_new_protected:Nn \stex_invoke_module:n {
1176   \stex_debug:nn{modules}{Invoking~module~#1}
1177   \peek_charcode_remove:NTF ! {
1178     \__stex_modules_invoke_uri:nN { #1 }
1179   } {
1180     \peek_charcode_remove:NTF ? {
1181       \__stex_modules_invoke_symbol:nn { #1 }
1182     } {
1183       \msg_error:nnn{stex}{error/syntax}{
1184         ?~or~!~expected~after~
1185         \c_backslash_str STEXModule{#1}
1186       }
1187     }
1188   }
1189 }
1190
1191 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1192   \str_set:Nn #2 { #1 }
1193 }
1194
1195 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1196   \stex_invoke_symbol:n{#1?#2}
1197 }

```

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

`\stex_activate_module:n`

```

1198 \cs_new_protected:Nn \stex_activate_module:n {
1199   \stex_debug:nn{modules}{Activating~module~#1}
1200   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1201     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1202     \prop_item:cn { c_stex_module_#1_prop } { content }
1203   }
1204 }

```

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

```

1205 </package>

```


Chapter 22

STEX -Module Inheritance Implementation

```
1206 <*package>
1207
1208 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1209
```

22.1 SMS Mode

```
1210 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1211 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1212 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1213 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1214
1215 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1216   \makeatletter
1217   \makeatother
1218   \ExplSyntaxOn
1219   \ExplSyntaxOff
1220 }
1221
1222 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1223   \symdef
1224   \importmodule
1225   \notation
1226   \symdecl
1227   \STEXexport
1228 }
1229
1230 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1231   \tl_to_str:n {
1232     module,
1233     @module
```

```

1234 }
1235 }

```

(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

```

```

1236 \bool_new:N \g__stex_smsmode_bool
1237 \bool_set_false:N \g__stex_smsmode_bool
1238 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1239   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1240 }

```

(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

```

```

1241 \bool_new:N \g__stex_smsmode_catcode_bool
1242 \bool_set_false:N \g__stex_smsmode_catcode_bool
1243 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1244   \bool_if:NTF \g__stex_smsmode_catcode_bool
1245   \prg_return_true: \prg_return_false:
1246 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1247 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1248   \stex_if_smsmode:T {
1249     \__stex_smsmode_if_catcodes:F {
1250       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1251       \exp_after:wN \char_gset_active_eq:NN
1252       \c_backslash_str \__stex_smsmode_cs:
1253       \tex_global:D \char_set_catcode_active:N \
1254       \tex_global:D \char_set_catcode_other:N $
1255       \tex_global:D \char_set_catcode_other:N ^
1256       \tex_global:D \char_set_catcode_other:N _
1257       \tex_global:D \char_set_catcode_other:N &
1258       \tex_global:D \char_set_catcode_other:N ##
1259     }
1260   }
1261 } \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.

```

1262 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1263   \__stex_smsmode_if_catcodes:T {
1264     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1265     \exp_after:wN \tex_global:D \exp_after:wN
1266     \char_set_catcode_escape:N \c_backslash_str
1267     \tex_global:D \char_set_catcode_math_toggle:N $
1268     \tex_global:D \char_set_catcode_math_superscript:N ^
1269     \tex_global:D \char_set_catcode_math_subscript:N _
1270     \tex_global:D \char_set_catcode_alignment:N &
1271     \tex_global:D \char_set_catcode_parameter:N ##
1272   }
1273 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1274 \cs_new_protected:Nn \stex_in_smsmode:nn {
1275   \vbox_set:Nn \l_tmpa_box {
1276     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1277     \bool_gset_true:N \g__stex_smsmode_bool
1278     \stex_smsmode_set_codes:
1279     #2
1280     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1281     \stex_if_smsmode:F {
1282       \__stex_smsmode_unset_codes:
1283     }
1284   }
1285   \box_clear:N \l_tmpa_box
1286 }

```

(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:

```

1287 \cs_new_protected:Nn \_stex_smsmode_cs: {
1288   \str_clear:N \l_tmpa_str
1289   \peek_analysis_map_inline:n {
1290     % #1: token (one expansion)
1291     % #2: charcode
1292     % #3 catcode
1293     \token_if_eq_charcode:NNTF ##3 B {
1294       % token is a letter
1295       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1296     } {
1297       \str_if_empty:NTF \l_tmpa_str {
1298         % we don't allow (or need) single non-letter CSs
1299         % for now
1300         \peek_analysis_map_break:
1301       }{
1302         \str_if_eq:onTF \l_tmpa_str { begin } {
1303           \peek_analysis_map_break:n {
1304             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1305           }
1306         } {
1307           \str_if_eq:onTF \l_tmpa_str { end } {
1308             \peek_analysis_map_break:n {
1309               \exp_after:wN \_stex_smsmode_checkend:n ##1
1310             }
1311           } {
1312             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1313             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1314               \g_stex_smsmode_allowedmacros_tl
1315               { \use:c{\l_tmpa_str} } {
1316               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1317               \peek_analysis_map_break:n {
1318                 \exp_after:wN \l_tmpa_tl ##1
1319               }

```

```

1320     } {
1321         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1322         \g_stex_smsmode_allowedmacros_escape_tl
1323         { \use:c{\l_tmpa_str} } {
1324             \__stex_smsmode_unset_codes:
1325             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1326             % TODO \__stex_smsmode_rescan_cs:
1327             \int_compare:nNnTF {##2} = {92} {
1328                 \peek_analysis_map_break:n {
1329                     \__stex_smsmode_unset_codes:
1330                     \__stex_smsmode_rescan_cs:
1331                 }
1332             } {
1333                 \peek_analysis_map_break:n {
1334                     \exp_after:wN \l_tmpa_tl ##1
1335                 }
1336             }
1337         } {
1338             \int_compare:nNnTF {##2} = {92} {
1339                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1340             } {
1341                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1342             }
1343         }
1344     }
1345 }
1346 }
1347 }
1348 }
1349 }
1350 }

```

(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:

```

1351 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1352     \str_clear:N \l_tmpb_str
1353     \peek_analysis_map_inline:n {
1354         \token_if_eq_charcode:NNTF ##3 B {
1355             % token is a letter
1356             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1357         } {
1358             \peek_analysis_map_break:n {
1359                 \exp_after:wN \use:c \exp_after:wN {
1360                     \exp_after:wN \l_tmpa_str\exp_after:wN
1361                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1362             }
1363         }
1364     }
1365 }

```

(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.

```

1366 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1367   \str_set:Nn \l_tmpa_str { #1 }
1368   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1369     \__stex_smsmode_unset_codes:
1370     \begin{#1}
1371   }
1372 }
```

(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.

```

1373 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1374   \str_set:Nn \l_tmpa_str { #1 }
1375   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1376     \end{#1}
1377   }
1378 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

1379 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1380 \cs_new_protected:Nn \stex_import_module_uri:nn {
1381   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1382   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1383   \str_if_empty:NT \l__stex_importmodule_archive_str {
1384     \prop_if_empty:NF \l_stex_current_repository_prop {
1385       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1386     }
1387   }
1388
1389   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1390   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1391   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1392
1393   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1394     \stex_modules_current_namespace:
1395     \str_if_empty:NF \l__stex_importmodule_path_str {
1396       \str_set:Nx \l_stex_module_ns_str {
1397         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1398       }
1399     }
1400   }{
1401     \stex_require_repository:n \l__stex_importmodule_archive_str
1402     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1403     \l_stex_module_ns_str
1404     \str_if_empty:NF \l__stex_importmodule_path_str {
1405       \str_set:Nx \l_stex_module_ns_str {
1406         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1407       }
1408     }
1409   }
```

```

1408     }
1409   }
1410 }

```

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

```

\l_stex_importmodule_name_str Store the return values of \stex_import_module_uri:nn.
\l_stex_importmodule_archive_str 1411 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1412 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1413 \str_new:N \l__stex_importmodule_path_str
1414 \str_new:N \g__stex_importmodule_file_str

```

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

```

\stex_import_require_module:nnnn {<ns>} {<archive-ID>} {<path>} {<name>}
1415 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1416   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1417
1418     % archive
1419     \str_set:Nx \l_tmpa_str { #2 }
1420     \str_if_empty:NTF \l_tmpa_str {
1421       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1422     } {
1423       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1424       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1425       \seq_put_right:Nn \l_tmpa_seq { source }
1426     }
1427
1428     % path
1429     \str_set:Nx \l_tmpb_str { #3 }
1430     \str_if_empty:NTF \l_tmpb_str {
1431       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1432
1433       \ltx@ifpackageloaded{babel} {
1434         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1435           { \language } \l_tmpb_str {
1436           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1437         }
1438       } {
1439         \str_clear:N \l_tmpb_str
1440       }
1441
1442       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1443       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1444         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1445       }{
1446         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1447         \IfFileExists{ \l_tmpa_str.tex }{
1448           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1449         }{
1450           % try english as default
1451           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1452           \IfFileExists{ \l_tmpa_str.en.tex }{
1453             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1454     }{
1455         \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1456     }
1457 }
1458 }
1459
1460 } {
1461     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1462     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1463
1464     \ltx@ifpackageloaded{babel} {
1465         \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1466             { \language } \l_tmpb_str {
1467             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1468         }
1469     } {
1470         \str_clear:N \l_tmpb_str
1471     }
1472
1473     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1474
1475     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1476     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1477         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1478     }{
1479         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1480         \IfFileExists{ \l_tmpa_str/#4.tex }{
1481             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1482         }{
1483             % try english as default
1484             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1485             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1486                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1487             }{
1488                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1489                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1490                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1491                 }{
1492                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1493                     \IfFileExists{ \l_tmpa_str.tex }{
1494                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1495                     }{
1496                         % try english as default
1497                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1498                         \IfFileExists{ \l_tmpa_str.en.tex }{
1499                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1500                         }{
1501                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1502                         }
1503                     }
1504                 }
1505             }
1506         }
1507     }

```

```

1508     }
1509
1510     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1511     \seq_clear:N \g_stex_modules_in_file_seq
1512     % \exp_args:Nnx \use:nn {
1513     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1514         \seq_clear:N \l_stex_all_modules_seq
1515         \prop_clear:N \l_stex_current_module_prop
1516         \str_set:Nx \l_tmpb_str { #2 }
1517         \str_if_empty:NF \l_tmpb_str {
1518             \stex_set_current_repository:n { #2 }
1519         }
1520         \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1521         \input { \g__stex_importmodule_file_str }
1522     }
1523     % }{
1524
1525     % }
1526     \prop_gput:Noo \g_stex_module_files_prop
1527     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1528     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1529
1530     \stex_if_module_exists:nF { #1 ? #4 } {
1531         \msg_error:nnn{stex}{error/unknownmodule}{
1532             #1?#4~(in~file~\g__stex_importmodule_file_str)
1533         }
1534     }
1535 }
1536 \stex_activate_module:n { #1 ? #4 }
1537 }

```

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

`\importmodule`

```

1538 \NewDocumentCommand \importmodule { O{} m } {
1539     \stex_import_module_uri:nn { #1 } { #2 }
1540     \stex_debug:nn{modules}{Importing~module:~
1541         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1542     }
1543     \stex_if_smsmode:F {
1544         \stex_import_require_module:nnnn
1545         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1546         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1547         \stex_annotate_invisible:nnn
1548         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1549     }
1550     \exp_args:Nx \stex_add_to_current_module:n {
1551         \stex_import_require_module:nnnn
1552         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1553         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1554     }
1555     \exp_args:Nx \stex_add_import_to_current_module:n {
1556         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1557     }

```



```

1558 \stex_smsmode_set_codes:
1559 }
1560 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1561 \NewDocumentCommand \usemodule { 0{} m } {
1562 \stex_if_smsmode:F {
1563 \stex_import_module_uri:nn { #1 } { #2 }
1564 \stex_import_require_module:nnnn
1565 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1566 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1567 \stex_annotate_invisible:nnn
1568 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1569 }
1570 \stex_smsmode_set_codes:
1571 }

```

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

```

1572 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1573 <*package>
1574
1575 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1576
```

Warnings and error messages

```
1577
```

23.1 Symbol Declarations

```
1578 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1579 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1580 \NewDocumentCommand \STEXsymbol { m } {
1581   \stex_get_symbol:n { #1 }
1582   \exp_args:No
1583   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1584 }
```

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

symdecl arguments:

```
1585 \keys_define:nn { stex / symdecl } {
1586   name      .str_set:N = \l_stex_symdecl_name_str ,
1587   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1588   args      .str_set:N = \l_stex_symdecl_args_str ,
1589   type      .tl_set:N  = \l_stex_symdecl_type_tl ,
1590   align     .str_set:N  = \l_stex_symdecl_align_str , % TODO(?)
1591   gfc       .str_set:N  = \l_stex_symdecl_gfc_str , % TODO(?)
1592   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1593   def       .tl_set:N  = \l_stex_symdecl_definiens_tl
1594 }
```

```

1595
1596 \bool_new:N \l_stex_symdecl_make_macro_bool
1597
1598 \cs_new_protected:Nn \__stex_symdecl_args:n {
1599   \str_clear:N \l_stex_symdecl_name_str
1600   \str_clear:N \l_stex_symdecl_args_str
1601   \bool_set_false:N \l_stex_symdecl_local_bool
1602   \tl_clear:N \l_stex_symdecl_type_tl
1603   \tl_clear:N \l_stex_symdecl_definiens_tl
1604
1605   \keys_set:nn { stex / symdecl } { #1 }
1606 }

```

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

```

1607
1608 \NewDocumentCommand \symdecl { s O{} m } {
1609   \__stex_symdecl_args:n { #2 }
1610   \IfBooleanTF #1 {
1611     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1612   } {
1613     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1614   }
1615   \stex_symdecl_do:n { #3 }
1616   \stex_smsmode_set_codes:
1617 }
1618 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1619 \cs_new_protected:Nn \stex_symdecl_do:n {
1620   \stex_if_in_module:F {
1621     % TODO throw error? some default namespace?
1622   }
1623
1624   \str_if_empty:NT \l_stex_symdecl_name_str {
1625     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1626   }
1627
1628   \prop_if_exist:cT { g_stex_symdecl_
1629     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1630     \prop_item:Nn \l_stex_current_module_prop {name} ?
1631     \l_stex_symdecl_name_str
1632     _prop
1633   }{
1634     % TODO throw error (beware of circular dependencies)
1635   }
1636
1637   \prop_clear:N \l_tmpa_prop
1638   \prop_put:Nnx \l_tmpa_prop { module } {
1639     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1640     \prop_item:Nn \l_stex_current_module_prop {name}
1641   }

```

```

1642 \seq_clear:N \l_tmpa_seq
1643 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1644 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1645 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1646 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1647
1648 \exp_args:No \stex_add_constant_to_current_module:n {
1649   \l_stex_symdecl_name_str
1650 }
1651
1652 % arity/args
1653 \int_zero:N \l_tmpb_int
1654
1655 \bool_set_true:N \l_tmpa_bool
1656 \str_map_inline:Nn \l_stex_symdecl_args_str {
1657   \token_case_meaning:NnF ##1 {
1658     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1659     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1660     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1661     {\tl_to_str:n a} {
1662       \bool_set_false:N \l_tmpa_bool
1663       \int_incr:N \l_tmpb_int
1664     }
1665     {\tl_to_str:n B} {
1666       \bool_set_false:N \l_tmpa_bool
1667       \int_incr:N \l_tmpb_int
1668     }
1669   }{
1670     \msg_set:nnn{stex}{error/wrongargs}{
1671       args~value~in~symbol~declaration~for~
1672       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1673       \prop_item:Nn \l_stex_current_module_prop {name} ?
1674       \l_stex_symdecl_name_str ~
1675       needs~to~be~
1676       i,~a,~b~or~B,~but~##1~given
1677     }
1678     \msg_error:nn{stex}{error/wrongargs}
1679   }
1680 }
1681 \bool_if:NTF \l_tmpa_bool {
1682   % possibly numeric
1683   \str_if_empty:NTF \l_stex_symdecl_args_str {
1684     \prop_put:Nnn \l_tmpa_prop { args } {}
1685     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1686   }{
1687     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1688     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1689     \str_clear:N \l_tmpa_str
1690     \int_step_inline:nn \l_tmpa_int {
1691       \str_put_right:Nn \l_tmpa_str i
1692     }
1693     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1694   }
1695 } {

```

```

1696     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1697     \prop_put:Nnx \l_tmpa_prop { arity }
1698       { \str_count:N \l_stex_symdecl_args_str }
1699   }
1700   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1701
1702
1703   % semantic macro
1704
1705   \bool_if:NT \l_stex_symdecl_make_macro_bool {
1706     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1707       \prop_item:Nn \l_tmpa_prop { module } ?
1708       \prop_item:Nn \l_tmpa_prop { name }
1709     } }
1710
1711     \bool_if:NF \l_stex_symdecl_local_bool {
1712       \exp_args:Nx \stex_add_to_current_module:n {
1713         \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1714           \prop_item:Nn \l_tmpa_prop { module } ?
1715           \prop_item:Nn \l_tmpa_prop { name }
1716         } }
1717       }
1718     }
1719   }
1720
1721   % add to all symbols
1722
1723   \bool_if:NF \l_stex_symdecl_local_bool {
1724     \exp_args:Nx \stex_add_to_current_module:n {
1725       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1726         \prop_item:Nn \l_tmpa_prop { module } ?
1727         \prop_item:Nn \l_tmpa_prop { name }
1728       }
1729     }
1730   }
1731
1732   \stex_debug:nn{symbols}{New~symbol:~
1733     \prop_item:Nn \l_tmpa_prop { module } ?
1734     \prop_item:Nn \l_tmpa_prop { name } ^^J
1735     Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1736     Args:~\prop_item:Nn \l_tmpa_prop { args }
1737   }
1738
1739   % circular dependencies require this:
1740
1741   \prop_if_exist:cF {
1742     g_stex_symdecl_
1743     \prop_item:Nn \l_tmpa_prop { module } ?
1744     \prop_item:Nn \l_tmpa_prop { name }
1745     _prop
1746   } {
1747     \prop_gset_eq:cN {
1748       g_stex_symdecl_
1749       \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1750     \prop_item:Nn \l_tmpa_prop { name }
1751     _prop
1752   } \l_tmpa_prop
1753 }
1754
1755 \stex_if_smsmode:TF {
1756   \bool_if:NF \l_stex_symdecl_local_bool {
1757     \exp_args:Nx \stex_add_to_sms:n {
1758       \prop_gset_from_keyval:cn {
1759         g_stex_symdecl_
1760         \prop_item:Nn \l_tmpa_prop { module } ?
1761         \prop_item:Nn \l_tmpa_prop { name }
1762         _prop
1763       } {
1764         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1765         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1766         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1767         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1768         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1769         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1770         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1771         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1772       }
1773       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1774         \prop_item:Nn \l_tmpa_prop { module } ?
1775         \prop_item:Nn \l_tmpa_prop { name }
1776       }
1777     }
1778   }
1779 }{
1780   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1781     \prop_item:Nn \l_tmpa_prop { module } ?
1782     \prop_item:Nn \l_tmpa_prop { name }
1783   }
1784   \stex_if_do_html:T {
1785     \stex_annotate_invisible:nnn {symdecl} {
1786       \prop_item:Nn \l_tmpa_prop { module } ?
1787       \prop_item:Nn \l_tmpa_prop { name }
1788     } {
1789       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1790       \stex_annotate_invisible:nnn{args}{}{
1791         \prop_item:Nn \l_tmpa_prop { args }
1792       }
1793       \stex_annotate_invisible:nnn{macroname}{}{#1}
1794       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1795         \stex_annotate_invisible:nnn{definiens}{}
1796         {\l_stex_symdecl_definiens_tl$}
1797       }
1798     }
1799   }
1800 }
1801 }

```

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

`\stex_get_symbol:n`

```
1802 \str_new:N \l_stex_get_symbol_uri_str
1803
1804 \cs_new_protected:Nn \stex_get_symbol:n {
1805   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1806     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1807   }{
1808     % argument is a string
1809     % is it a command name?
1810     \cs_if_exist:cTF { #1 }{
1811       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1812       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1813       \str_if_empty:NNTF \l_tmpa_str {
1814         \exp_args:Nx \cs_if_eq:NNTF {
1815           \tl_head:N \l_tmpa_tl
1816         } \stex_invoke_symbol:n {
1817           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1818         }{
1819           \__stex_symdecl_get_symbol_from_string:n { #1 }
1820         }
1821       } {
1822         \__stex_symdecl_get_symbol_from_string:n { #1 }
1823       }
1824     }{
1825       % argument is not a command name
1826       \__stex_symdecl_get_symbol_from_string:n { #1 }
1827       % \l_stex_all_symbols_seq
1828     }
1829   }
1830 }
1831
1832 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1833   \str_set:Nn \l_tmpa_str { #1 }
1834   \bool_set_false:N \l_tmpa_bool
1835   \stex_if_in_module:T {
1836     \prop_get:NnN \l_stex_current_module_prop
1837     { constants } \l_tmpa_seq
1838     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1839       \bool_set_true:N \l_tmpa_bool
1840       \str_set:Nx \l_stex_get_symbol_uri_str {
1841         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1842         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1843       }
1844     }
1845   }
1846   \bool_if:NF \l_tmpa_bool {
1847     \tl_set:Nn \l_tmpa_tl {
1848       \msg_set:nnn{stex}{error/unknownsymbol}{
1849         No~symbol~#1~found!
1850       }
1851     }
1852     \msg_error:nn{stex}{error/unknownsymbol}
1853   }
1854   \str_set:Nn \l_tmpa_str { #1 }
1855   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1855 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1856   \str_set:Nn \l_tmpb_str { ##1 }
1857   \str_if_eq:eeT { \l_tmpa_str } {
1858     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1859   } {
1860     \seq_map_break:n {
1861       \tl_set:Nn \l_tmpa_tl {
1862         \str_set:Nn \l_stex_get_symbol_uri_str {
1863           ##1
1864         }
1865       }
1866     }
1867   }
1868 }
1869 \l_tmpa_tl
1870 }
1871 }
1872
1873 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1874   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1875   { \tl_tail:N \l_tmpa_tl }
1876   \tl_if_single:NTF \l_tmpa_tl {
1877     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1878       \exp_after:wN \str_set:Nn \exp_after:wN
1879       \l_stex_get_symbol_uri_str \l_tmpa_tl
1880     }{
1881       % TODO
1882       % tail is not a single group
1883     }
1884   }{
1885     % TODO
1886     % tail is not a single group
1887   }
1888 }

```

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

23.2 Notations

```

1889 <@@=stex_notation>
1890 notation arguments:
1891 \keys_define:nn { stex / notation } {
1892   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1893   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1894   prec .str_set_x:N = \l__stex_notation_prec_str ,
1895   op .tl_set:N = \l__stex_notation_op_tl ,
1896   unknown .code:n = \str_set:Nx
1897     \l__stex_notation_variant_str \l_keys_key_str
1898 }
1899 \cs_new_protected:Nn \__stex_notation_args:n {
1900   \str_clear:N \l__stex_notation_lang_str
1901   \str_clear:N \l__stex_notation_variant_str

```



```

1902 \str_clear:N \l__stex_notation_prec_str
1903 \tl_clear:N \l__stex_notation_op_tl
1904
1905 \keys_set:nn { stex / notation } { #1 }
1906 }

```

\notation

```

1907 \NewDocumentCommand \notation { 0{ } m } {
1908   \__stex_notation_args:n { #1 }
1909   \tl_clear:N \l_stex_symdecl_definiens_tl
1910   \stex_get_symbol:n { #2 }
1911   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1912 }
1913 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

1914 \cs_new_protected:Nn \stex_notation_do:nn {
1915   \prop_set_eq:Nc \l_tmpa_prop {
1916     g_stex_symdecl_ #1 _prop
1917   }
1918
1919   \prop_clear:N \l_tmpb_prop
1920   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1921   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1922   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1923
1924   % precedences
1925   \seq_clear:N \l_tmpb_seq
1926   \exp_args:NNno
1927   \str_if_empty:NTF \l__stex_notation_prec_str {
1928     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1929     \int_compare:nNnTF \l_tmpa_str = 0 {
1930       \exp_args:NNnx
1931       \prop_put:Nno \l_tmpb_prop { opprec }
1932       { \neginfprec }
1933     }{
1934       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1935     }
1936   } {
1937     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1938       \exp_args:NNnx
1939       \prop_put:Nno \l_tmpb_prop { opprec }
1940       { \neginfprec }
1941       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1942       \int_step_inline:nn { \l_tmpa_str } {
1943         \exp_args:NNx
1944         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1945       }
1946     }{
1947       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1948       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1949         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1950         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1951         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1952         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1953         \seq_map_inline:Nn \l_tmpa_seq {
1954             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1955         }
1956     }
1957     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1958 }{
1959     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1960     \int_compare:nNnTF \l_tmpa_str = 0 {
1961         \exp_args:NNnx
1962         \prop_put:Nno \l_tmpb_prop { opprec }
1963         { \infprec }
1964     }{
1965         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1966     }
1967 }
1968 }
1969 }
1970
1971 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1972 \int_step_inline:nn { \l_tmpa_str } {
1973     \seq_pop_left:NnF \l_tmpa_seq \l_tmpb_str {
1974         \exp_args:NNx
1975         \seq_put_right:Nn \l_tmpb_seq {
1976             \prop_item:Nn \l_tmpb_prop { opprec }
1977         }
1978     }
1979 }
1980
1981 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1982 \tl_clear:N \l_tmpa_tl
1983
1984 \int_compare:nNnTF \l_tmpa_str = 0 {
1985     \exp_args:NNe
1986     \cs_set:Npn \l__stex_notation_macrocode_cs {
1987         \_stex_term_math_oms:nnnn { #1 }
1988         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1989         { \prop_item:Nn \l_tmpb_prop { opprec } }
1990         { \exp_not:n { #2 } }
1991     }
1992     \__stex_notation_final:
1993 }{
1994     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1995     \str_if_in:NnTF \l_tmpb_str b {
1996         \exp_args:Nne \use:nn
1997         {
1998             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1999             \cs_set:Npn \l_tmpa_str { {
2000                 \_stex_term_math_omb:nnnn { #1 }
2001                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2002                 { \prop_item:Nn \l_tmpb_prop { opprec } }
2003                 { \exp_not:n { #2 } }
2004             }
2005         }
2006     }

```

```

2005   }{
2006     \str_if_in:NnTF \l_tmpb_str B {
2007       \exp_args:Nne \use:nn
2008       {
2009         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2010         \cs_set:Npn \l_tmpa_str } { {
2011           \stex_term_math_omb:nnnn { #1 }
2012           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2013           { \prop_item:Nn \l_tmpb_prop { opprec } }
2014           { \exp_not:n { #2 } }
2015         } }
2016       }{
2017         \exp_args:Nne \use:nn
2018         {
2019           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2020           \cs_set:Npn \l_tmpa_str } { {
2021             \stex_term_math_oma:nnnn { #1 }
2022             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2023             { \prop_item:Nn \l_tmpb_prop { opprec } }
2024             { \exp_not:n { #2 } }
2025           } }
2026         }
2027       }
2028
2029       \int_zero:N \l_tmpa_int
2030       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2031       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2032       \__stex_notation_arguments:
2033     }
2034   }

```

(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

```

2035 \cs_new_protected:Nn \__stex_notation_arguments: {
2036   \int_incr:N \l_tmpa_int
2037   \str_if_empty:NnTF \l_tmpa_str {
2038     \__stex_notation_final:
2039   }{
2040     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2041     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2042     \str_if_eq:NnTF \l_tmpb_str a {
2043       \__stex_notation_argument_assoc:n
2044     }{
2045       \str_if_eq:NnTF \l_tmpb_str B {
2046         \__stex_notation_argument_assoc:n
2047       }{
2048         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2049         \tl_put_right:Nx \l_tmpa_tl {
2050           { \stex_term_math_arg:nnn
2051             { \int_use:N \l_tmpa_int }
2052             { \l_tmpb_str }
2053             { ####\int_use:N \l_tmpa_int }
2054           }

```

```

2055     }
2056     \__stex_notation_arguments:
2057   }
2058 }
2059 }
2060 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2061 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2062   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2063   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2064   \tl_put_right:Nx \l_tmpa_tl {
2065     { \stex_term_math_assoc_arg:nnnn
2066       { \int_use:N \l_tmpa_int }
2067       { \l_tmpb_str }
2068       \exp_args:No \exp_not:n
2069       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2070       { ####\int_use:N \l_tmpa_int }
2071     }
2072   }
2073   \__stex_notation_arguments:
2074 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2075 \cs_new_protected:Nn \__stex_notation_final: {
2076   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2077   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2078   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2079   \exp_args:Nne \use:nn
2080   {
2081     \cs_generate_from_arg_count:cNnn {
2082       stex_notation_ \l_tmpa_str \c_hash_str
2083       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2084       _cs
2085     }
2086     \cs_gset:Npn \l_tmpb_str } { {
2087       \exp_after:wN \exp_after:wN \exp_after:wN
2088       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2089       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2090     } }
2091
2092   \tl_if_empty:NF \l__stex_notation_op_tl {
2093     \cs_gset:cpx {
2094       stex_op_notation_ \l_tmpa_str \c_hash_str
2095       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2096       _cs
2097     } {
2098       \stex_term_oms:nnn {
2099         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2100         \l__stex_notation_lang_str

```

```

2101     }{
2102         \l_tmpa_str
2103     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2104 }
2105 }
2106
2107
2108
2109 \stex_debug:nn{symbols}{
2110     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2111     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2112     Operator~precedence:~
2113     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2114     Argument~precedences:~
2115     \seq_use:Nn \l_tmpa_seq {,~}^^J
2116     Notation: \cs_meaning:c {
2117         stex_notation_ \l_tmpa_str \c_hash_str
2118         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2119         _cs
2120     }
2121 }
2122
2123 \prop_gset_eq:cN {
2124     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2125     \c_hash_str \l__stex_notation_lang_str _prop
2126 } \l_tmpb_prop
2127
2128 \exp_args:Nx
2129 \stex_add_to_current_module:n {
2130     \prop_get:cnN {
2131         g_stex_symdecl_
2132         \prop_item:Nn \l_tmpb_prop { symbol }
2133         _prop
2134     } { notations } \exp_not:N \l_tmpa_seq
2135     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2136         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2137     }
2138     \prop_put:cno {
2139         g_stex_symdecl_
2140         \prop_item:Nn \l_tmpb_prop { symbol }
2141         _prop
2142     } { notations } \exp_not:N \l_tmpa_seq
2143 }
2144
2145 \stex_if_smsmode:TF {
2146     \stex_smsmode_set_codes:
2147     \exp_args:Nx \stex_add_to_sms:n {
2148         \prop_gset_from_keyval:cn {
2149             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2150             \c_hash_str \l__stex_notation_lang_str _prop
2151         } {
2152             symbol = \prop_item:Nn \l_tmpb_prop { symbol } ,
2153             language = \prop_item:Nn \l_tmpb_prop { language } ,
2154             variant = \prop_item:Nn \l_tmpb_prop { variant } ,

```

```

2155         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2156         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2157     }
2158 }
2159 }{
2160   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2161   \seq_put_right:Nx \l_tmpa_seq {
2162     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2163   }
2164   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2165   \prop_set_eq:cN {
2166     g_stex_symdecl_ \l_tmpa_str _prop
2167   } \l_tmpa_prop
2168
2169   % HTML annotations
2170   \stex_if_do_html:T {
2171     \stex_annotate_invisible:nnn { notation }
2172     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2173       \stex_annotate_invisible:nnn { notationfragment }
2174       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2175       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2176       \stex_annotate_invisible:nnn { precedence }
2177       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2178         \seq_use:Nn \l_tmpa_seq { x }
2179       }{}
2180
2181       \int_zero:N \l_tmpa_int
2182       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2183       \tl_clear:N \l_tmpa_tl
2184       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2185         \int_incr:N \l_tmpa_int
2186         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2187         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2188         \str_if_eq:VnTF \l_tmpb_str a {
2189           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2190             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2191             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2192           } }
2193         }{
2194           \str_if_eq:VnTF \l_tmpb_str B {
2195             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2196               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2197               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2198             } }
2199           }{
2200             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2201               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2202             } }
2203           }
2204         }
2205       }
2206       \stex_annotate_invisible:nnn { notationcomp }{}{
2207         $ \exp_args:Nno \use:nn { \use:c {
2208           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2209         \c_hash_str \l__stex_notation_variant_str
2210         \c_hash_str \l__stex_notation_lang_str _cs
2211     } } { \l_tmpa_tl } $
2212 }
2213 }
2214 }
2215 }
2216 }

```

(End definition for `__stex_notation_final:`.)

\symdef

```

2217 \keys_define:nn { stex / symdef } {
2218   name .str_set_x:N = \l_stex_symdecl_name_str ,
2219   local .bool_set:N = \l_stex_symdecl_local_bool ,
2220   args .str_set_x:N = \l_stex_symdecl_args_str ,
2221   type .tl_set:N = \l_stex_symdecl_type_tl ,
2222   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2223   op .tl_set:N = \l__stex_notation_op_tl ,
2224   lang .str_set_x:N = \l__stex_notation_lang_str ,
2225   variant .str_set_x:N = \l__stex_notation_variant_str ,
2226   prec .str_set_x:N = \l__stex_notation_prec_str ,
2227   unknown .code:n = \str_set:Nx
2228     \l__stex_notation_variant_str \l_keys_key_str
2229 }
2230
2231 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2232   \str_clear:N \l_stex_symdecl_name_str
2233   \str_clear:N \l_stex_symdecl_args_str
2234   \bool_set_false:N \l_stex_symdecl_local_bool
2235   \tl_clear:N \l_stex_symdecl_type_tl
2236   \tl_clear:N \l_stex_symdecl_definiens_tl
2237   \str_clear:N \l__stex_notation_lang_str
2238   \str_clear:N \l__stex_notation_variant_str
2239   \str_clear:N \l__stex_notation_prec_str
2240   \tl_clear:N \l__stex_notation_op_tl
2241
2242   \keys_set:nn { stex / symdef } { #1 }
2243 }
2244
2245 \NewDocumentCommand \symdef { 0{} m } {
2246   \__stex_notation_symdef_args:n { #1 }
2247   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2248   \stex_symdecl_do:n { #2 }
2249   \exp_args:Nx \stex_notation_do:nn {
2250     \prop_item:Nn \l_tmpa_prop { module } ?
2251     \prop_item:Nn \l_tmpa_prop { name }
2252   }
2253 }
2254 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2255 \endpackage

```

Chapter 24

STEX -Terms Implementation

```
2256 <*package>
2257
2258 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2259
2260 <@@=stex_terms>
2261
2262   Warnings and error messages
2263   \msg_new:nnn{stex}{error/nonotation}{
2264     Symbol~#1~invoked,~but~has~no~notation#2!
2265   }
2266   \msg_new:nnn{stex}{error/notationarg}{
2267     Error~in~parsing~notation~#1
2268   }
2269
```

24.1 Symbol Invocations

Arguments:

```
2268 \keys_define:nn { stex / terms } {
2269   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2270   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2271   unknown .code:n = \str_set:Nx
2272     \l__stex_terms_variant_str \l_keys_key_str
2273 }
2274
2275 \cs_new_protected:Nn \__stex_terms_args:n {
2276   \str_clear:N \l__stex_terms_lang_str
2277   \str_clear:N \l__stex_terms_variant_str
2278   \str_clear:N \l__stex_terms_prec_str
2279   \tl_clear:N \l__stex_terms_op_tl
2280
2281   \keys_set:nn { stex / terms } { #1 }
2282 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro


```

2283 \cs_new_protected:Nn \stex_invoke_symbol:n {
2284   \if_mode_math:
2285     \exp_after:wN \__stex_terms_invoke_math:n
2286   \else:
2287     \exp_after:wN \__stex_terms_invoke_text:n
2288   \fi: { #1 }
2289 }

```

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

`__stex_terms_invoke_math:n`

```

2290 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2291   \peek_charcode_remove:NTF ! {
2292     \peek_charcode:NTF [ {
2293       \__stex_terms_invoke_op:nw { #1 }
2294     }{
2295       \__stex_terms_invoke_op:nw { #1 } []
2296     }
2297   }{
2298     \peek_charcode_remove:NTF * {
2299       \__stex_terms_invoke_text:n { #1 }
2300     }{
2301       \peek_charcode:NTF [ {
2302         \__stex_terms_invoke_math:nw { #1 }
2303       }{
2304         \__stex_terms_invoke_math:nw { #1 } []
2305       }
2306     }
2307   }
2308 }

```

(End definition for `__stex_terms_invoke_math:n`.)

`__stex_terms_invoke_op:nw`

```

2309 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2310   \__stex_terms_args:n { #2 }
2311   \cs_if_exist:cTF {
2312     stex_op_notation_ #1 \c_hash_str
2313     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2314   }{
2315     \csname stex_op_notation_ #1 \c_hash_str
2316     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2317   \endcsname
2318   }{
2319     % TODO throw error
2320   }
2321 }

```

(End definition for `__stex_terms_invoke_op:nw`.)

`__stex_terms_invoke_math:nw`

```

2322 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2323   \__stex_terms_args:n { #2 }
2324   \prop_set_eq:Nc \l_tmpa_prop {
2325     g_stex_symdecl_ #1 _prop

```

```

2326 }
2327 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2328 \seq_if_empty:NTF \l_tmpa_seq {
2329   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2330 } {
2331   \seq_if_in:NxTF \l_tmpa_seq
2332     { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2333     \use:c{
2334       stex_notation_ #1 \c_hash_str
2335       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2336     }_cs
2337   }
2338   }{
2339     \str_if_empty:NTF \l__stex_terms_variant_str {
2340       \str_if_empty:NTF \l__stex_terms_lang_str {
2341         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2342         \use:c{
2343           stex_notation_ #1 \c_hash_str \l_tmpa_str
2344         }_cs
2345       }
2346     }{
2347       \msg_error:nn{stex}{error/nonotation}{#1}{
2348         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2349       }
2350     }
2351   }{
2352     \msg_error:nn{stex}{error/nonotation}{#1}{
2353       ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2354     }
2355   }
2356 }
2357 }
2358 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2359 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2360   \peek_charcode_remove:NTF ! {
2361     \stex_term_custom:nn { #1 } { }
2362   }{
2363     \prop_set_eq:Nc \l_tmpa_prop {
2364       g_stex_symdecl_ #1 _prop
2365     }
2366     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2367     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2368   }
2369 }

```

(End definition for `__stex_terms_invoke_text:n`.)

24.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2370 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2371 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2372 \int_new:N \l__stex_terms_downprec
2373 \int_set_eq:NN \l__stex_terms_downprec \infprec

(End definition for \infprec, \neginfprec, and \l__stex_terms_downprec. These variables are docu-
mented on page 28.)

Bracketing:

\l__stex_terms_left_bracket_str
\l__stex_terms_right_bracket_str
2374 \tl_set:Nn \l__stex_terms_left_bracket_str (
2375 \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
2376 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2377   \bool_if:NTF \l__stex_terms_brackets_done_bool {
2378     \bool_set_false:N \l__stex_terms_brackets_done_bool
2379     #2
2380   } {
2381     \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2382       \bool_if:NTF \l__stex_inarray_bool { #2 }{
2383         \stex_debug:nn{dobrackets}{Here! \number#1 > \number\l__stex_terms_downprec; \detokenize{
2384           \dobrackets { #2 }
2385         }
2386       }{ #2 }
2387     }
2388   }

(End definition for \__stex_terms_maybe_brackets:nn.)

\dobrackets
2389 \bool_new:N \l__stex_terms_brackets_done_bool
2390 %\RequirePackage{scalerel}
2391 \cs_new_protected:Npn \dobrackets #1 {
2392   %\ThisStyle{\if D\m@switch
2393   %   \exp_args:Nnx \use:nn
2394   %   { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2395   %   { \exp_not:N\right\l__stex_terms_right_bracket_str }
2396   % \else
2397   \exp_args:Nnx \use:nn
2398   {
2399     \bool_set_true:N \l__stex_terms_brackets_done_bool
2400     \int_set:Nn \l__stex_terms_downprec \infprec
2401     \l__stex_terms_left_bracket_str
2402     #1
2403   }
2404   {
2405     \bool_set_false:N \l__stex_terms_brackets_done_bool
2406     \l__stex_terms_right_bracket_str
2407     \int_set:Nn \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2408   }
2409   %\fi}
2410 }

```

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

`\withbrackets`

```

2411 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2412   \exp_args:Nnx \use:nn
2413   {
2414     \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2415     \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2416     #3
2417   }
2418   {
2419     \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2420     {\l__stex_terms_left_bracket_str}
2421     \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str
2422     {\l__stex_terms_right_bracket_str}
2423   }
2424 }

```

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

`\STEXinvisible`

```

2425 \cs_new_protected:Npn \STEXinvisible #1 {
2426   \stex_annotate_invisible:n { #1 }
2427 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2428 \cs_new_protected:Nn \_stex_term_oms:nnn {
2429   \stex_annotate:nnn{ OMID }{ #2 }{
2430     \stex_highlight_term:nn { #1 } { #3 }
2431   }
2432 }
2433
2434 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2435   \__stex_terms_maybe_brackets:nn { #3 }{
2436     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2437   }
2438 }

```

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

`_stex_term_math_oma:nnnn`

```

2439 \cs_new_protected:Nn \_stex_term_oma:nnn {
2440   \stex_annotate:nnn{ OMA }{ #2 }{
2441     \stex_highlight_term:nn { #1 } { #3 }
2442   }
2443 }
2444
2445 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2446   \__stex_terms_maybe_brackets:nn { #3 }{
2447     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2448   }
2449 }

```

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

`_stex_term_math_omb:nnnn`

```

2450 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2451   \stex_annotate:nnn{ OMBIND }{ #2 }{
2452     \stex_highlight_term:nn { #1 } { #3 }
2453   }
2454 }
2455
2456 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2457   \_stex_terms_maybe_brackets:nn { #3 }{
2458     \stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2459   }
2460 }

```

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

`_stex_term_math_arg:nnn`

```

2461 \cs_new_protected:Nn \_stex_term_arg:nn {
2462   \stex_unhighlight_term:n {
2463     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2464   }
2465 }
2466 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2467   \exp_args:Nnx \use:nn
2468     { \int_set:Nn \l__stex_terms_downprec { #2 }
2469       \stex_term_arg:nn { #1 }{ #3 }
2470     }
2471   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2472   }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2473 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2474   \clist_set:Nn \l_tmpa_clist{ #4 }
2475   \int_compare:nNnTF { \clist_count:N \l_tmpa_clist } < 2 {
2476     \tl_set:Nn \l_tmpa_tl { #4 }
2477   }{
2478     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2479     \clist_reverse:N \l_tmpa_clist
2480     \clist_pop:NN \l_tmpa_clist \l_tmpa_tl
2481
2482     \clist_map_inline:Nn \l_tmpa_clist {
2483       \exp_args:NNNo \exp_args:NNo \tl_set:No \l_tmpa_tl {
2484         \exp_args:Nno
2485           \l_tmpa_cs { ##1 } \l_tmpa_tl
2486       }
2487     }
2488
2489   }
2490   \exp_args:Nnno
2491   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2492 }

```

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

`\stex_term_custom:nn`

```

2493 \cs_new_protected:Nn \stex_term_custom:nn {
2494   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2495   \str_set:Nn \l_tmpa_str { #2 }
2496   \tl_clear:N \l_tmpa_tl
2497   \int_zero:N \l_tmpa_int
2498   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2499   \__stex_terms_custom_loop:
2500 }

```

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

`__stex_terms_custom_loop:`

```

2501 \cs_new_protected:Nn \__stex_terms_custom_loop: {
2502   \bool_set_false:N \l_tmpa_bool
2503   \bool_while_do:nn {
2504     \str_if_eq_p:ee X {
2505       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2506     }
2507   }{
2508     \int_incr:N \l_tmpa_int
2509   }
2510
2511   \peek_charcode:NTF [ {
2512     % notation/text component
2513     \__stex_terms_custom_component:w
2514   } {
2515     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2516       % all arguments read => finish
2517       \__stex_terms_custom_final:
2518     } {
2519       % arguments missing
2520       \peek_charcode_remove:NTF * {
2521         % invisible, specific argument position or both
2522         \peek_charcode:NTF [ {
2523           % visible specific argument position
2524           \__stex_terms_custom_arg:wn
2525         } {
2526           % invisible
2527           \peek_charcode_remove:NTF * {
2528             % invisible specific argument position
2529             \__stex_terms_custom_arg_inv:wn
2530           } {
2531             % invisible next argument
2532             \__stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2533           }
2534         }
2535       } {
2536         % next normal argument
2537         \__stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2538       }
2539     }

```

```

2540 }
2541 }

(End definition for \_stex_terms_custom_loop:.)

```

_stex_terms_custom_arg_inv:wn

```

2542 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2543   \bool_set_true:N \l_tmpa_bool
2544   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2545 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2546 \cs_new_protected:Npn \_stex_terms_custom_arg:wn [ #1 ] #2 {
2547   \str_set:Nx \l_tmpb_str {
2548     \str_item:Nn \l_tmpa_str { #1 }
2549   }
2550   \str_case:VnTF \l_tmpb_str {
2551     { X } {
2552       \msg_error:nnn{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2553     }
2554     { i } { \_stex_terms_custom_set_X:n { #1 } }
2555     { b } { \_stex_terms_custom_set_X:n { #1 } }
2556     { a } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2557     { B } { \_stex_terms_custom_set_X:n { #1 } } % TODO ?
2558   }{}{
2559     \msg_error:nnn{stex}{error/notationarg}{\l_stex_terms_custom_uri}
2560   }
2561
2562   \bool_if:nTF \l_tmpa_bool {
2563     \tl_put_right:Nx \l_tmpa_tl {
2564       \stex_annotate_invisible:n {
2565         \stex_term_arg:nn { \int_eval:n { #1 } }
2566         \exp_not:n { { #2 } }
2567       }
2568     }
2569   } {
2570     \tl_put_right:Nx \l_tmpa_tl {
2571       \stex_term_arg:nn { \int_eval:n { #1 } }
2572       \exp_not:n { { #2 } }
2573     }
2574   }
2575
2576   \_stex_terms_custom_loop:
2577 }

```

(End definition for _stex_terms_custom_arg:wn.)

_stex_terms_custom_set_X:n

```

2578 \cs_new_protected:Nn \_stex_terms_custom_set_X:n {
2579   \str_set:Nx \l_tmpa_str {
2580     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2581     X
2582     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }

```

```

2583 }
2584 }

(End definition for \_stex_terms_custom_set_X:n.)

```

_stex_terms_custom_component:

```

2585 \cs_new_protected:Npn \_stex_terms_custom_component:w [ #1 ] {
2586   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2587   \_stex_terms_custom_loop:
2588 }

```

(End definition for _stex_terms_custom_component:.)

_stex_terms_custom_final:

```

2589 \cs_new_protected:Nn \_stex_terms_custom_final: {
2590   \int_compare:nNnTF \l_tmpb_int = 0 {
2591     \exp_args:Nnno \_stex_term_oms:nnn
2592   }{
2593     \str_if_in:NnTF \l_tmpa_str {b} {
2594       \exp_args:Nnno \_stex_term_ombind:nnn
2595     } {
2596       \exp_args:Nnno \_stex_term_oma:nnn
2597     }
2598   }
2599   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2600 }

```

(End definition for _stex_terms_custom_final:.)

\symref

\symname

```

2601 \NewDocumentCommand \symref { m m }{
2602   \let\compemph_uri_prev:\compemph@uri
2603   \let\compemph@uri\symrefemph@uri
2604   \STEXsymbol{#1}! [#2]
2605   \let\compemph@uri\compemph_uri_prev:
2606 }
2607
2608 \keys_define:nn { stex / symname } {
2609   post      .str_set_x:N    = \l_stex_symname_post_str
2610 }
2611
2612 \cs_new_protected:Nn \stex_symname_args:n {
2613   \str_clear:N \l_stex_symname_post_str
2614   \keys_set:nn { stex / symname } { #1 }
2615 }
2616
2617 \NewDocumentCommand \symname { 0{} m }{
2618   \stex_symname_args:n { #1 }
2619   \stex_get_symbol:n { #2 }
2620   \str_set:Nx \l_tmpa_str {
2621     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2622   }
2623   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {-}
2624
2625   \let\compemph_uri_prev:\compemph@uri

```



```

2626 \let\compemph@uri\symrefemph@uri
2627 \exp_args:NNx \use:nn
2628 \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2629   \l_tmpa_str \l_stex_symname_post_str
2630 ] }
2631 \let\compemph@uri\compemph_uri_prev:
2632 }

```

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

24.3 Notation Components

```

2633 <@@=stex_notationcomps>

```

`\stex_highlight_term:nn`

```

2634
2635 \str_new:N \l__stex_notationcomps_highlight_uri_str
2636 \cs_new_protected:Nn \stex_highlight_term:nn {
2637   \exp_args:Nnx
2638   \use:nn {
2639     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2640     #2
2641   } {
2642     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2643       { \l__stex_notationcomps_highlight_uri_str }
2644   }
2645 }
2646
2647 \cs_new_protected:Nn \stex_unhighlight_term:n {
2648   % \latexml_if:TF {
2649   %   #1
2650   % } {
2651   %   \scalatex_if:TF {
2652   %     #1
2653   %   } {
2654     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2655   % }
2656   % }
2657 }

```

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

```

\comp
\compemph@uri
\compemph
\defemph
\defemph@uri
\symrefemph
\symrefemph@uri
2658 \cs_new_protected:Npn \comp #1 {
2659   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
2660     \scalatex_if:TF {
2661       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
2662     }{
2663       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2664     }
2665   }
2666 }
2667
2668 \cs_new_protected:Npn \compemph@uri #1 #2 {

```

```

2669     \compemph{ #1 }
2670 }
2671
2672
2673 \cs_new_protected:Npn \compemph #1 {
2674     \textcolor{blue}{#1}
2675 }
2676
2677 \cs_new_protected:Npn \defemph@uri #1 #2 {
2678     \defemph{#1}
2679 }
2680
2681 \cs_new_protected:Npn \defemph #1 {
2682     \textbf{#1}
2683 }
2684
2685 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2686     \symrefemph{#1}
2687 }
2688
2689 \cs_new_protected:Npn \symrefemph #1 {
2690     \textbf{#1}
2691 }

```

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

\ellipses

```

2692 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2693 \bool_new:N \l_stex_inarray_bool
2694 \bool_set_false:N \l_stex_inarray_bool
2695 \NewDocumentCommand \parray { m m } {
2696     \begingroup
2697     \bool_set_true:N \l_stex_inarray_bool
2698     \begin{array}{#1}
2699         #2
2700     \end{array}
2701     \endgroup
2702 }
2703
2704 \NewDocumentCommand \prmatrix { m } {
2705     \begingroup
2706     \bool_set_true:N \l_stex_inarray_bool
2707     \begin{matrix}
2708         #1
2709     \end{matrix}
2710     \endgroup
2711 }
2712
2713 \def \parrayline #1 #2 {
2714     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2715 }

```

```

2716
2717 \def \parraylineh #1 #2 {
2718   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2719 }
2720
2721 \def \parraycell #1 {
2722   #1 \bool_if:NT \l_stex_inarray_bool {&}
2723 }

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

2724 \endpackage

```

Chapter 25

STEX -Structural Features Implementation

```
2725 <*package>
2726
2727 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2728
2729 <@@=stex_features>
      Warnings and error messages
2730
```

25.1 The feature environment

structural@feature

```
2731
2732 \NewDocumentEnvironment{structural@feature}{ m m m }{
2733   \stex_if_in_module:F {
2734     \msg_set:nnn{stex}{error/nomodule}{
2735       Structural~Feature~has~to~occur~in~a~module:\\
2736       Feature~#2~of~type~#1\\
2737       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2738     }
2739     \msg_error:nn{stex}{error/nomodule}
2740   }
2741
2742   \str_set:Nx \l_stex_module_name_str {
2743     \prop_item:Nn \l_stex_current_module_prop
2744       { name } / #2 - feature
2745   }
2746
2747   \str_set:Nx \l_stex_module_ns_str {
2748     \prop_item:Nn \l_stex_current_module_prop
2749       { ns }
2750   }
2751
```

```

2752
2753 \str_clear:N \l_tmpa_str
2754 \seq_clear:N \l_tmpa_seq
2755 \tl_clear:N \l_tmpa_tl
2756 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2757   origname = #2,
2758   name     = \l_stex_module_name_str ,
2759   ns       = \l_stex_module_ns_str ,
2760   imports  = \exp_not:o { \l_tmpa_seq } ,
2761   constants = \exp_not:o { \l_tmpa_seq } ,
2762   content  = \exp_not:o { \l_tmpa_tl } ,
2763   file     = \exp_not:o { \g_stex_currentfile_seq } ,
2764   lang     = \l_stex_module_lang_str ,
2765   sig      = \l_tmpa_str ,
2766   meta     = \l_tmpa_str ,
2767   feature  = #1 ,
2768 }
2769
2770 \stex_if_smsmode:TF {
2771   \stex_smsmode_set_codes:
2772 } {
2773   \begin{stex_annotate_env}{ feature:#1 }{}
2774   \stex_annotate_invisible:nnn{header}{}{ #3 }
2775 }
2776 }{
2777   \str_set:Nx \l_tmpa_str {
2778     c_stex_feature_
2779     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2780     \prop_item:Nn \l_stex_current_module_prop { name }
2781     _prop
2782   }
2783   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2784   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2785   \stex_if_smsmode:TF {
2786     \exp_args:Nx \stex_add_to_sms:n {
2787       \prop_gset_from_keyval:cn {
2788         c_stex_feature_
2789         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2790         \prop_item:Nn \l_stex_current_module_prop { name }
2791         _prop
2792       } {
2793         origname = #2,
2794         name     = \prop_item:cn { \l_tmpa_str } { name } ,
2795         ns       = \prop_item:cn { \l_tmpa_str } { ns } ,
2796         imports  = \prop_item:cn { \l_tmpa_str } { imports } ,
2797         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2798         content  = \prop_item:cn { \l_tmpa_str } { content } ,
2799         file     = \prop_item:cn { \l_tmpa_str } { file } ,
2800         lang     = \prop_item:cn { \l_tmpa_str } { lang } ,
2801         sig      = \prop_item:cn { \l_tmpa_str } { sig } ,
2802         meta     = \prop_item:cn { \l_tmpa_str } { meta } ,
2803         feature  = \prop_item:cn { \l_tmpa_str } { feature }
2804       }
2805     }

```

```

2806 } {
2807     \end{stex_annotate_env}
2808 }
2809 }
2810

```

25.2 Features

structure

```

2811
2812 \prop_new:N \l_stex_all_structures_prop
2813
2814 \keys_define:nn { stex / features / structure } {
2815     name .str_set_x:N = \l__stex_features_structure_name_str ,
2816 }
2817
2818 \cs_new_protected:Nn \__stex_features_structure_args:n {
2819     \str_clear:N \l__stex_features_structure_name_str
2820     \keys_set:nn { stex / features / structure } { #1 }
2821 }
2822
2823 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2824 % \__stex_features_structure_args:n { ##1 }
2825 % \str_if_empty:NT \l__stex_features_structure_name_str {
2826 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2827 % }
2828 %} {
2829 %
2830 %}
2831
2832 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2833     \__stex_features_structure_args:n { #1 }
2834     \str_if_empty:NT \l__stex_features_structure_name_str {
2835         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2836     }
2837     \exp_args:Nnnx
2838     \begin{structural@feature}{ structure }
2839         { \l__stex_features_structure_name_str }{}
2840         \seq_clear:N \l_tmpa_seq
2841         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2842
2843     }{
2844         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2845         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2846         \str_set:Nx \l_tmpa_str {
2847             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2848             \prop_item:Nn \l_stex_current_module_prop { name }
2849         }
2850         \seq_map_inline:Nn \l_tmpa_seq {
2851             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2852         }
2853         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2854         \exp_args:Nnx

```

```

2855 \AddToHookNext { env / mathstructure / after }{
2856 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2857 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2858 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2859 \STEXexport {
2860 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2861 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2862 {\l_tmpa_str}
2863 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2864 {#2}{\l_tmpa_str}
2865 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2866 % \prop_item:Nn \l_stex_current_module_prop { origname },
2867 % \l_tmpa_str
2868 % }
2869 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2870 % #2,\l_tmpa_str
2871 % }
2872 % \tl_set:cx { #2 } {
2873 % \stex_invoke_structure:n { \l_tmpa_str }
2874 }
2875 }
2876
2877 \end{structural@feature}
2878 % \g_stex_last_feature_prop
2879 }

```

\instantiate

```

2880 \seq_new:N \l__stex_features_structure_field_seq
2881 \str_new:N \l__stex_features_structure_field_str
2882 \str_new:N \l__stex_features_structure_def_tl
2883 \prop_new:N \l__stex_features_structure_prop
2884 \NewDocumentCommand \instantiate { m O{} m }{
2885 \stex_smsmode_set_codes:
2886 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2887 \prop_set_eq:Nc \l__stex_features_structure_prop {
2888 c_stex_feature_\l_tmpa_str _prop
2889 }
2890 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2891 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2892 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2893 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2894 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2895 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2896 {!} \l_tmpa_tl
2897 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2898 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2899 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2900 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2901 }{
2902 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2903 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2904 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2905 \l_tmpa_tl
2906 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2907         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2908         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2909     }{
2910         \tl_clear:N \l_tmpb_tl
2911     }
2912 }
2913 }{
2914     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2915     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2916         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2917         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2918         \tl_clear:N \l_tmpa_tl
2919     }{
2920         % TODO throw error
2921     }
2922 }
2923 % \l_tmpa_str: name
2924 % \l_tmpa_tl: definiens
2925 % \l_tmpb_tl: notation
2926 \tl_if_empty:NT \l__stex_features_structure_field_str {
2927     % TODO throw error
2928 }
2929 \str_clear:N \l_tmpb_str
2930
2931 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2932 \seq_map_inline:Nn \l_tmpa_seq {
2933     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2934     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2935     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2936         \seq_map_break:n {
2937             \str_set:Nn \l_tmpb_str { ####1 }
2938         }
2939     }
2940 }
2941 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2942     \l_tmpb_str
2943
2944 \tl_if_empty:NTF \l_tmpb_tl {
2945     \tl_if_empty:NF \l_tmpa_tl {
2946         \exp_args:Nx \use:n {
2947             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2948         }
2949     }
2950 }{
2951     \tl_if_empty:NTF \l_tmpa_tl {
2952         \exp_args:Nx \use:n {
2953             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2954         }
2955     }{
2956         \exp_args:Nx \use:n {
2957             \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2958             \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2959         }
2960     }

```



```

2961     }
2962   }
2963   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2964   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2965   % #3/\l_stex_features_structure_field_str
2966   % \par
2967   % \expandafter\present\csname
2968   %   g_stex_symdecl_
2969   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2970   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2971   %   #3/\l_stex_features_structure_field_str
2972   %   _prop
2973   % \endcsname
2974 }
2975
2976 \tl_clear:N \l__stex_features_structure_def_tl
2977
2978 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2979 \seq_map_inline:Nn \l_tmpa_seq {
2980   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2981   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2982   \exp_args:Nx \use:n {
2983     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2984
2985     }
2986   }
2987
2988   \prop_if_exist:cF {
2989     g_stex_symdecl_
2990     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2991     \prop_item:Nn \l_stex_current_module_prop {name} ?
2992     #3/\l_tmpa_str
2993     _prop
2994   }{
2995     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2996     \l_tmpb_str
2997     \exp_args:Nx \use:n {
2998       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2999     }
3000   }
3001 }
3002
3003 \symdecl*[type={\STEXsymbol{module-type}}{
3004   \stex_term_math_oms:nnnn {
3005     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3006     \prop_item:Nn \l__stex_features_structure_prop {name}
3007     }{}{0}{}
3008   }{}{#3}
3009
3010 % TODO: -> sms file
3011
3012 \tl_set:cx{ #3 }{
3013   \stex_invoke_structure:nnn {
3014     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

3015     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
3016   } {
3017     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
3018     \prop_item:Nn \l__stex_features_structure_prop {name}
3019   }
3020 }
3021
3022 }

```

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

\stex_invoke_structure:nnn

```

3023 % #1: URI of the instance
3024 % #2: URI of the instantiated module
3025 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3026   \tl_if_empty:nTF{ #3 }{
3027     \prop_set_eq:Nc \l__stex_features_structure_prop {
3028       c_stex_feature_ #2 _prop
3029     }
3030     \tl_clear:N \l_tmpa_tl
3031     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3032     \seq_map_inline:Nn \l_tmpa_seq {
3033       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3034       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3035       \cs_if_exist:cT {
3036         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3037       }{
3038         \tl_if_empty:NF \l_tmpa_tl {
3039           \tl_put_right:Nn \l_tmpa_tl {,}
3040         }
3041         \tl_put_right:Nx \l_tmpa_tl {
3042           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3043         }
3044       }
3045     }
3046     \exp_args:No \mathstrut \l_tmpa_tl
3047   }{
3048     \stex_invoke_symbol:n{#1/#3}
3049   }
3050 }

```

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

```

3051 </package>

```

Chapter 26

STEX -Statements Implementation

```
3052 <*package>
3053
3054 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3055
3056 \protected\def\ignorespacesandpars{
3057   \begingroup\catcode13=10\relax
3058   \@ifnextchar\par{
3059     \endgroup\expandafter\ignorespacesandpars\@gobble
3060   }{
3061     \endgroup
3062   }
3063 }
3064
3065 <@@=stex_statements>
3066
3067   Warnings and error messages
3068
3069 \def\titleemph#1{\textbf{#1}}
3070
3071 symboldoc
3072
3073 \NewDocumentEnvironment{symboldoc}{m}{
3074   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3075   \seq_clear:N \l_tmpb_seq
3076   \seq_map_inline:Nn \l_tmpa_seq {
3077     \str_if_eq:nnF{ ##1 }{}{
3078       \stex_get_symbol:n { ##1 }
3079       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3080         \l_stex_get_symbol_uri_str
3081       }
3082     }
3083   }
3084   \par
3085   \exp_args:Nnnx
3086   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3087 }{
```

```

3083 \end{stex_annotate_env}
3084 }

3085 \seq_new:N \g_stex_statements_patched_seq
3086
3087 \cs_new_protected:Nn \stex_statements_set_patched:n {
3088   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3089 }
3090
3091 \cs_new_protected:Nn \stex_statements_patch:nn {
3092   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3093     \AddToHook{begindocument}{
3094       \cs_if_exist:cTF{end#1}{
3095         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3096         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3097       }{
3098         \NewDocumentEnvironment{#1}{0{}}{
3099           \use:c{__stex_statements_#2_begin:n}{ }
3100         }{
3101           \use:c{__stex_statements_#2_end:}
3102         }
3103       }
3104     }
3105   }
3106 }

```

26.1 Definitions

definition

```

3107
3108 \NewDocumentCommand \definiendum { 0{ } m m } {
3109   \stex_get_symbol:n { #2 }
3110   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3111   \scalatex_if:TF {
3112     \stex_annotate:nnn { \definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3113   } {
3114     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3115   }
3116 }
3117 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3118 \NewDocumentCommand \definame { 0{ } m } {
3119   % TODO: root
3120   \stex_get_symbol:n { #2 }
3121   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3122   \str_set:Nx \l_tmpa_str {
3123     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3124   }
3125   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3126   \scalatex_if:TF {
3127     \stex_annotate:nnn { \definendum } { \l_stex_get_symbol_uri_str } {
3128       \l_tmpa_str
3129     }
3130   } {

```

```

3131     \defemph@uri {
3132         \l_tmpa_str
3133     } { \l_stex_get_symbol_uri_str }
3134 }
3135 }
3136 \stex_deactivate_macro:Nn \definame {definition-environments}
3137
3138 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3139     \stex_reactivate_macro:N \definiendum
3140     \stex_reactivate_macro:N \definame
3141     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3142     \seq_clear:N \l_tmpb_seq
3143     \seq_map_inline:Nn \l_tmpa_seq {
3144         \str_if_eq:nnF{ ##1 }{}{
3145             \stex_get_symbol:n { ##1 }
3146             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3147                 \l_stex_get_symbol_uri_str
3148             }
3149         }
3150     }
3151     \stex_smsmode_set_codes:
3152     \exp_args:Nnnx
3153     \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3154 }
3155
3156 \cs_new_protected:Nn \__stex_statements_defi_end: {
3157     \end{stex_annotate_env}
3158 }

```

Hook:

```

3159 \stex_statements_patch:nn{definition}{defi}

inline:

3160 \NewDocumentCommand \inlinedef { m } {
3161     \begingroup
3162     \stex_reactivate_macro:N \definiendum
3163     \stex_reactivate_macro:N \definame
3164     \stex_ref_new_doc_target:n{
3165         #1
3166     }
3167 }

```

26.2 Assertions

assertion

```

3168 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3169     \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3170     \seq_clear:N \l_tmpb_seq
3171     \seq_map_inline:Nn \l_tmpa_seq {
3172         \str_if_eq:nnF{ ##1 }{}{
3173             \stex_get_symbol:n { ##1 }
3174             \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3175                 \l_stex_get_symbol_uri_str

```

```

3176     }
3177   }
3178 }
3179 \titleemph{Assertion}~
3180 \stex_smsmode_set_codes:
3181 \exp_args:Nnnx
3182 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3183 }
3184
3185 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3186   \end{stex_annotate_env}
3187 }

```

Hook:

```

3188 \stex_statements_patch:nn{assertion}{assertion}

```

inline:

```

3189 \NewDocumentCommand \inlineass { m } {
3190   \begingroup
3191   \stex_ref_new_doc_target:n{
3192     #1
3193   }
3194 }

```

theorem

```

3195 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3196   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3197   \seq_clear:N \l_tmpb_seq
3198   \seq_map_inline:Nn \l_tmpa_seq {
3199     \str_if_eq:nnF{ ##1 }{}{
3200       \stex_get_symbol:n { ##1 }
3201       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3202         \l_stex_get_symbol_uri_str
3203       }
3204     }
3205   }
3206   \titleemph{Theorem}~
3207   \stex_smsmode_set_codes:
3208   \exp_args:Nnnx
3209   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3210 }
3211
3212 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3213   \end{stex_annotate_env}
3214 }

```

Hook:

```

3215 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3216 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3217   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3218   \seq_clear:N \l_tmpb_seq

```

```

3219 \seq_map_inline:Nn \l_tmpa_seq {
3220 \str_if_eq:nnF{ ##1 }{}{
3221   \stex_get_symbol:n { ##1 }
3222   \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3223     \l_stex_get_symbol_uri_str
3224   }
3225 }
3226 }
3227 \titleemph{Lemma}~
3228 \stex_smsmode_set_codes:
3229 \exp_args:Nnnx
3230 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3231 }
3232
3233 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3234   \end{stex_annotate_env}
3235 }

```

Hook:

```

3236 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3237 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3238   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3239   \seq_clear:N \l_tmpb_seq
3240   \seq_map_inline:Nn \l_tmpa_seq {
3241     \str_if_eq:nnF{ ##1 }{}{
3242       \stex_get_symbol:n { ##1 }
3243       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3244         \l_stex_get_symbol_uri_str
3245       }
3246     }
3247   }
3248   \titleemph{Axiom}~
3249   \stex_smsmode_set_codes:
3250   \exp_args:Nnnx
3251   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3252 }
3253
3254 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3255   \end{stex_annotate_env}
3256 }

```

Hook:

```

3257 \stex_statements_patch:nn{axiom}{axiom}

```

26.3 Examples

example

```

3258 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3259   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3260   \seq_clear:N \l_tmpb_seq

```

```

3261 \seq_map_inline:Nn \l_tmpa_seq {
3262   \str_if_eq:nnF{ ##1 }{}{
3263     \stex_get_symbol:n { ##1 }
3264     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3265       \l_stex_get_symbol_uri_str
3266     }
3267   }
3268 }
3269 \titleemph{Example}~
3270 \stex_smsmode_set_codes:
3271 \exp_args:Nnnx
3272 \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3273 }
3274
3275 \cs_new_protected:Nn \__stex_statements_example_end: {
3276   \end{stex_annotate_env}
3277 }

```

Hook:

```

3278 \stex_statements_patch:nn{example}{example}

inline:
3279 \NewDocumentCommand \inlineex { m } {
3280   \begingroup
3281     \stex_ref_new_doc_target:n{
3282       #1
3283     }
3284   \endgroup
3285 }

```

26.4 OMText

```

3285 \keys_define:nn { stex / omtex } {
3286   id      .str_set_x:N = \l_stex_omtext_id_str ,
3287   title   .tl_set:N   = \l_stex_omtext_title_tl ,
3288   type    .tl_set_x:N  = \l_stex_omtext_type_tl ,
3289   for     .tl_set_x:N  = \l_stex_omtext_for_tl ,
3290   from    .tl_set_x:N  = \l_stex_omtext_from_tl ,
3291   start   .tl_set:N    = \l_stex_omtext_start_tl ,
3292 }
3293 \cs_new_protected:Nn \stex_omtext_args:n {
3294   \tl_clear:N \l_stex_omtext_title_tl
3295   \tl_clear:N \l_stex_omtext_start_tl
3296   \keys_set:nn { stex / omtex } { #1 }
3297 }
3298 \newif\if@in@omtext\@in@omtextfalse
3299 \NewDocumentEnvironment {omtext} { 0{} } {
3300   \stex_omtext_args:n { #1 }
3301   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3302     \tl_if_empty:NF \l_stex_omtext_title_tl {
3303       \titleemph{\l_stex_omtext_title_tl}:~
3304     }
3305   }{
3306     \titleemph{\l_stex_omtext_start_tl}~

```



```

3307 }
3308 \@in@omtexttrue
3309
3310 \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3311 \stex_smsmode_set_codes:
3312 \ignorespacesandpars
3313 }{}
3314 \end{package}

```

Chapter 27

The Implementation

27.1 Package Options

We declare some switches which will modify the behavior according to the package options. Generally, an option `xxx` will just set the appropriate switches to true (otherwise they stay false).¹⁰

```
3315 <*package>
3316 <@@=stex_sproof>
3317
3318 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3319
```

27.2 Proofs

We first define some keys for the proof environment.

```
3320 \keys_define:nn { stex / spf } {
3321   id          .str_set:N = \l__stex_sproof_spf_id_str,
3322   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3323   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3324   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3325   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3326   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3327   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3328   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3329   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3330   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3331 }
3332 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3333   \str_clear:N \l__stex_sproof_spf_id_str
3334   \tl_clear:N \l__stex_sproof_spf_display_tl
3335   \tl_clear:N \l__stex_sproof_spf_for_tl
3336   \tl_clear:N \l__stex_sproof_spf_from_tl
3337   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3338   \tl_clear:N \l__stex_sproof_spf_type_tl
3339   \tl_clear:N \l__stex_sproof_spf_title_tl
```

¹⁰EDNOTE: need an implementation for L^AT_EX_ML

```

3340 \tl_clear:N \l__stex_sproof_spf_continues_tl
3341 \tl_clear:N \l__stex_sproof_spf_functions_tl
3342 \tl_clear:N \l__stex_sproof_spf_method_tl
3343 \keys_set:nn { stex / spf }{ #1 }
3344 }

```

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

```

3345 \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.

```

3346 \newcount\count_ten
3347 \newenvironment{pst@with@label}[1]{
3348   \edef\pst@label{#1}
3349   \advance\count_ten by 1\relax
3350   \count_ten=1
3351 }{
3352   \advance\count_ten by -1\relax
3353 }

```

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

```

3354 \def\the@pst@label{
3355   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3356 }

```

(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.

```

3357 \keys_define:nn { stex / pstlabel }{
3358   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3359   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3360   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3361 }
3362 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

3363 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3364 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3365 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3366 }
3367 \__stex_sproof_pstlabel_args:n {}
3368 \newcommand\setpstlabelstyle[1]{
3369   \__stex_sproof_pstlabel_args:n {#1}
3370 }
3371 \newcommand\setpstlabelstyledefault{%
3372   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3373 }

```

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

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

```

3374 \ExplSyntaxOff
3375 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3376 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3377 \def\pst@make@label@short#1#2{#2}
3378 \def\pst@make@label@empty#1#2{}
3379 \ExplSyntaxOn
3380 \def\pstlabelstyle#1{%
3381   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3382 }%
3383 \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.

```

3384 \def\next@pst@label{%
3385   \global\advance\count\count10 by 1%
3386 }%

```

(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

```

3387 \def\sproof@box{
3388   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3389 }
3390 \def\spf@proofend{\sproof@box}
3391 \def\sproofend{
3392   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3393     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3394   }
3395 }
3396 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3397 \def\spf@proofsketch@kw{Proof Sketch}
3398 \def\spf@proof@kw{Proof}
3399 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3400 \cs_if_exist:NT \bbl@loaded {
3401   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3402   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3403     \input{proof-ngerman.lda}
3404   }
3405   \clist_if_in:NnT \l_tmpa_clist {finnish}{
3406     \input{proof-finnish.lda}
3407   }
3408   \clist_if_in:NnT \l_tmpa_clist {french}{
3409     \input{proof-french.lda}
3410   }
3411   \clist_if_in:NnT \l_tmpa_clist {russian}{
3412     \input{proof-russian.lda}
3413   }
3414 }
3415

```

spfsketch

```

3416 \newcommand\spfsketch[2][]{
3417   \__stex_sproof_spf_args:n{#1}
3418   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3419     \titleemph{
3420       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3421         \spf@proofsketch@kw
3422       }{
3423         \l__stex_sproof_spf_type_tl
3424       }
3425     }:
3426   }
3427   {-#2}
3428   %\sref@label@id{this \ifx\spf@type\empty\spf@proofsketch@kw\else\spf@type\fi}
3429   \sproofend
3430 }

```

(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

```

3431 \newenvironment{spfeq}[2][]{
3432   \__stex_sproof_spf_args:n{#1}
3433   %\sref@target
3434   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3435     \titleemph{
3436       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3437         \spf@proof@kw
3438       }{
3439         \l__stex_sproof_spf_type_tl
3440       }
3441     }:

```

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

¹²EDNOTE: document above

```

3442 }
3443 {~#2}
3444 \begin{displaymath}\begin{array}{rcll}
3445 }{
3446 \end{array}\end{displaymath}
3447 }

```

(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.

```

3448 \newenvironment{spf@proof}[2][]{
3449 \__stex_sproof_spf_args:n{#1}
3450 %\sref@target
3451 \count_ten=10
3452 \par\noindent
3453 \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3454 \titleemph{
3455 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3456 \spf@proof@kw
3457 }{
3458 \l__stex_sproof_spf_type_tl
3459 }
3460 } :
3461 }
3462 {~#2}
3463 %\sref@label@id{this \ifx\spf@type\empty\spf@proof@kw\else\spf@type\fi}
3464 \def\pst@label{}
3465 \newcount\pst@count% initialize the labeling mechanism
3466 \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3467 }{
3468 \end{pst@with@label}\end{description}
3469 }
3470 \newenvironment{sproof}[2][]{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3471 \newenvironment{sProof}[2][]{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3472 \newcommand\spfidea[2][]{
3473 \__stex_sproof_spf_args:n{#1}
3474 \titleemph{
3475 \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3476 \l__stex_sproof_spf_type_tl
3477 } :
3478 }~#2
3479 \sproofend
3480 }

```

(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

```

3481 \newenvironment{spfstep}[1][]{
3482   \_stex_sproof_spf_args:n{#1}
3483   \@in@omtexttrue
3484   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3485     \item[\the@pst@label]
3486   }
3487   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3488     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3489   }
3490   %\sref@label@id{\pst@label}
3491   \ignorespacesandpars
3492 }{
3493   \next@pst@label\ignorespacesandpars
3494 }

```

sproofcomment

```

3495 \newenvironment{sproofcomment}[1][]{
3496   \_stex_sproof_spf_args:n{#1}
3497   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3498     \item[\the@pst@label]
3499   }
3500 }{
3501   \next@pst@label
3502 }

```

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.

```

3503 \newenvironment{subproof}[2][]{
3504   \_stex_sproof_spf_args:n{#1}
3505   \def\@test{#2}
3506   \ifx\@test\empty\else
3507     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3508       \item[\the@pst@label]
3509     }{#2}
3510   \fi
3511   \begin{pst@with@label}{\pst@label,\number\count_ten}
3512 }{
3513   \end{pst@with@label}\next@pst@label
3514 }

```

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

```

3515 \newenvironment{spfcases}[2][]{
3516   \def\@test{#1}
3517   \ifx\@test\empty
3518     \begin{subproof}[method=by-cases]{#2}
3519   \else
3520     \begin{subproof}[#1,method=by-cases]{#2}
3521   \fi
3522 }{

```

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

```

3523 \end{subproof}
3524 }

```

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

```

3525 \newenvironment{spfcase}[2] [] {
3526   \__stex_sproof_spf_args:n{#1}
3527   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3528     \item[\the@pst@label]
3529   }
3530   \def\@test{#2}
3531   \ifx\@test\@empty
3532   \else
3533     {\titleemph{#2}:~}
3534   \fi
3535   \begin{pst@with@label}{\pst@label,\number\count_ten}
3536 }{
3537   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3538     \sproofend
3539   }
3540   \end{pst@with@label}
3541   \next@pst@label
3542 }

```

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

```

3543 \newcommand\spfcasesketch[3] [] {
3544   \__stex_sproof_spf_args:n{#1}
3545   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3546     \item[\the@pst@label]
3547   }
3548   \def\@test{#2}
3549   \ifx\@test\@empty
3550   \else
3551     {\titleemph{#2}:~}
3552   \fi#3
3553   \next@pst@label
3554 }%

```

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.

```

3555 \keys_define:nn { stex / just }{
3556   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3557   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3558   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3559   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3560 }

```

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

```
3561 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3562 \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.

```
3563 \newcommand\justarg[2] [] {#2}
```

```
3564 \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

```
3565 <*package>
3566
3567 %%%%%%%%%% others.dtx %%%%%%%%%%
3568
3569 <@@=stex_others>
    Warnings and error messages
3570 % None

\MSC Math subject classifier

3571 \NewDocumentCommand \MSC {m} {
3572 % TODO
3573 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3574 \@ifpackageloaded{tikzinput}{
3575 \RequirePackage{stex-tikzinput}
3576 }{}
3577 </package>
```

Chapter 29

STEX -Metatheory Implementation

```
3578 <*package>
3579 <@@=stex_modules>
3580
3581 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3582
3583 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3584 \begingroup
3585 \stex_module_setup:nn{
3586   ns=\c_stex_metatheory_ns_str,
3587   meta=NONE
3588 }{Metatheory}
3589 \stex_reactivate_macro:N \symdecl
3590 \stex_reactivate_macro:N \notation
3591 \stex_reactivate_macro:N \symdef
3592 \ExplSyntaxOff
3593 \csname stex_suppress_html:n\endcsname{
3594   % is-a (a:A, a \in A, a is an A, etc.)
3595   \symdecl[args=ai]{isa}
3596   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3597   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3598   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3599
3600   % bind (\forall, \Pi, \lambda etc.)
3601   \symdecl[args=Bi]{bind}
3602   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3603   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3604   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3605
3606   % dummy variable
3607   \symdecl{dummyvar}
3608   \notation[underscore]{dummyvar}{\comp\_}
3609   \notation[dot]{dummyvar}{\comp\cdot}
3610   \notation[dash]{dummyvar}{\comp{\rm --}}
3611
3612   %fromto (function space, Hom-set, implication etc.)
```

```

3613 \symdecl[args=ai]{fromto}
3614 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3615 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3616
3617 % mapto (lambda etc.)
3618 %\symdecl[args=Bi]{mapto}
3619 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3620 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3621 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3622
3623 % function/operator application
3624 \symdecl[args=ia]{apply}
3625 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3626 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3627
3628 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3629 \symdecl{collection}
3630 \notation[U]{collection}{\comp{\mathcal{U}}{}}
3631 \notation[set]{collection}{\comp{\textsf{Set}}{}}
3632
3633 % sequences
3634 \symdecl[args=1]{seqtype}
3635 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3636
3637 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3638 \notation[ui]{sequence-index}{#1^{\#2}}
3639
3640 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3641 %\notation[ui]{sequence-from-to}{#1^{\#2}\comp{\,\ellipses,}#1^{\#3}}
3642 % ^ superceded by \aseqfromto and \livar/\uivar
3643
3644 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3645 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
3646 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
3647
3648 % letin (‘‘let’’, local definitions, variable substitution)
3649 \symdecl[args=bii]{letin}
3650 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3651 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3652 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3653
3654 % structures
3655 \symdecl*[args=1]{module-type}
3656 \notation{module-type}{\mathtt{MOD} #1}
3657 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3658 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3659
3660 }
3661 \ExplSyntaxOn
3662 \stex_add_to_current_module:n{
3663   \let\nappa\apply
3664   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3665   \def\livar{\csname sequence-index\endcsname[li]}
3666   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3667     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3668     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3669     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3670   }
3671   \__stex_modules_end_module:
3672   \endgroup
3673 </package>

```

Chapter 30

Tikzinput Implementation

```
3674 <*package>
3675
3676 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3677
3678 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3679 \RequirePackage{l3keys2e}
3680
3681 \keys_define:nn { tikzinput } {
3682   image .bool_set:N = \c_tikzinput_image_bool,
3683   image .default:n = false ,
3684   unknown .code:n = {}
3685 }
3686
3687 \ProcessKeysOptions { tikzinput }
3688
3689 \bool_if:NTF \c_tikzinput_image_bool {
3690   \RequirePackage{graphicx}
3691
3692   \providecommand\usetikzlibrary[]{}
3693   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3694 }{
3695   \RequirePackage{tikz}
3696   \RequirePackage{standalone}
3697
3698   \newcommand \tikzinput [2] [] {
3699     \setkeys{Gin}{#1}
3700     \ifx \Gin@ewidth \Gin@exclamation
3701       \ifx \Gin@eheight \Gin@exclamation
3702         \input { #2 }
3703       \else
3704         \resizebox{!}{ \Gin@eheight }{
3705           \input { #2 }
3706         }
3707       \fi
3708     \else
3709       \ifx \Gin@eheight \Gin@exclamation
3710         \resizebox{ \Gin@ewidth }{!}{
3711           \input { #2 }
```

```

3712     }
3713     \else
3714         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3715             \input { #2 }
3716         }
3717     \fi
3718 \fi
3719 }
3720 }
3721
3722 \newcommand \ctikzinput [2] [] {
3723     \begin{center}
3724         \tikzinput [1] {#2}
3725     \end{center}
3726 }
3727
3728 \@ifpackageloaded{stex}{
3729     \RequirePackage{stex-tikzinput}
3730 }{}
3731
3732 </package>
3733 <*stex>
3734 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3735 \RequirePackage{stex}
3736 \RequirePackage{tikzinput}
3737
3738 \newcommand\mhtikzinput [2] [] {%
3739     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3740     \stex_in_repository:nn\Gin@mhrepos{
3741         \tikzinput [1]{\mhpath{##1}{#2}}
3742     }
3743 }
3744 \newcommand\cmhtikzinput [2] [] {\begin{center}\mhtikzinput [1] {#2}\end{center}}
3745 </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.

```
3746 \*cls)
3747 \@@=document_structure)
3748 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3749 \RequirePackage{l3keys2e,expl-keystr-compat}
```

31.2 Class Options

To initialize the `omdoc` class, we declare and process the necessary options using the `kvoptions` package for key/value options handling. For `omdoc.cls` this is quite simple. We have options `report` and `book`, which set the `\omdoc@cls@class` macro and pass on the macro to `omdoc.sty` for further processing.

`\omdoc@cls@class`

```
3750 \keys_define:nn{ document-structure / pkg }{
3751   class      .str_set_x:N = \c_document_structure_class_str,
3752   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3753   report     .code:n      = {
3754     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3755     \str_set:Nn \c_document_structure_class_str {report}
3756   },
3757   book       .code:n      = {
3758     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3759     \str_set:Nn \c_document_structure_class_str {book}
3760   },
3761   bookpart   .code:n      = {
3762     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3763     \str_set:Nn \c_document_structure_class_str {book}
3764     \str_set:Nn \c_document_structure_topsect_str {chapter}
3765   },
```



```

3766 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3767 unknown     .code:n      = {
3768   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3769 }
3770 }
3771 \ProcessKeysOptions{ document-structure / pkg }
3772 \str_if_empty:NT \c_document_structure_class_str {
3773   \str_set:Nn \c_document_structure_class_str {article}
3774 }
3775 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3776   {\c_document_structure_class_str}
3777

```

31.3 Beefing up the document environment

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

```

3778 \RequirePackage{omdoc}
3779 \bool_if:NF \c_document_structure_minimal_bool {
3780   \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

```

3781 \keys_define:nn { document-structure / document }{
3782   id .str_set_x:N = \c_document_structure_document_id_str
3783 }
3784 \let\__document_structure_orig_document=\document
3785 \renewcommand{\document}[1][]{
3786   \keys_set:nn{ document-structure / document }{ #1 }
3787   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3788   \__document_structure_orig_document
3789 }

```

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

```

3790 }
3791 \</cls>

```

31.4 Implementation: OMDoc Package

```

3792 \*package>
3793 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3794 \RequirePackage{expl-keys-compact,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

```

3795
3796 \keys_define:nn{ document-structure / pkg }{
3797   class      .str_set_x:N = \c_document_structure_class_str,
3798   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3799   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3800 }
3801 \ProcessKeysOptions{ document-structure / pkg }
3802 \str_if_empty:NT \c_document_structure_class_str {
3803   \str_set:Nn \c_document_structure_class_str {article}
3804 }
3805 \str_if_empty:NT \c_document_structure_topsect_str {
3806   \str_set:Nn \c_document_structure_topsect_str {section}
3807 }

```

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

```

3808 \RequirePackage{xspace}
3809 \RequirePackage{comment}
3810 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3811 \@ifpackageloaded{babel}{
3812   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3813   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3814     \input{omdoc-ngerman.ldf}
3815   }
3816 }{}
3817 %\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.

```

3818 \int_new:N \l_document_structure_section_level_int
3819 \str_case:VnF \c_document_structure_topsect_str {
3820   {part}{
3821     \int_set:Nn \l_document_structure_section_level_int {0}
3822   }
3823   {chapter}{
3824     \int_set:Nn \l_document_structure_section_level_int {1}
3825   }
3826 }{
3827   \str_case:VnF \c_document_structure_class_str {
3828     {book}{
3829       \int_set:Nn \l_document_structure_section_level_int {0}
3830     }
3831     {report}{
3832       \int_set:Nn \l_document_structure_section_level_int {0}
3833     }
3834   }{
3835     \int_set:Nn \l_document_structure_section_level_int {2}
3836   }
3837 }

```

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

```
3838 \def\current@section@level{document}%
3839 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3840 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3841 \cs_new_protected:Npn \skipomgroup {
3842   \ifcase\l_document_structure_section_level_int
3843   \or\stepcounter{part}
3844   \or\stepcounter{chapter}
3845   \or\stepcounter{section}
3846   \or\stepcounter{subsection}
3847   \or\stepcounter{subsubsection}
3848   \or\stepcounter{paragraph}
3849   \or\stepcounter{subparagraph}
3850   \fi
3851 }
```

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

`blindomgroup`

```
3852 \newcommand\at@begin@blindomgroup[1]{%
3853 \newenvironment{blindomgroup}
3854 {
3855   \int_incr:N\l_document_structure_section_level_int
3856   \at@begin@blindomgroup\l_document_structure_section_level_int
3857 }{}}
```

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

```
3858 \newcommand\omgroup@nonum[2]{
3859   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3860   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3861 }
```

(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.

```
3862 \newcommand\omgroup@num[2]{
```

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

```

3863 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3864   \@nameuse{#1}{#2}
3865 }{
3866   \cs_if_exist:NTF\rdfmata@sectioning{
3867     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3868   }{
3869     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3870   }
3871 }
3872 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3873 }

```

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

omgroup

```

3874 \keys_define:nn { document-structure / omgroup }{
3875   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3876   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
3877   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
3878   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3879   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
3880   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
3881   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
3882   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
3883   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
3884   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
3885 }
3886 \cs_new_protected:Nn \__document_structure_omgroup_args:n {
3887   \str_clear:N \l__document_structure_omgroup_id_str
3888   \str_clear:N \l__document_structure_omgroup_date_str
3889   \clist_clear:N \l__document_structure_omgroup_creators_clist
3890   \clist_clear:N \l__document_structure_omgroup_contributors_clist
3891   \tl_clear:N \l__document_structure_omgroup_srccite_tl
3892   \tl_clear:N \l__document_structure_omgroup_type_tl
3893   \tl_clear:N \l__document_structure_omgroup_short_tl
3894   \tl_clear:N \l__document_structure_omgroup_display_tl
3895   \tl_clear:N \l__document_structure_omgroup_intro_tl
3896   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
3897   \keys_set:nn { document-structure / omgroup } { #1 }
3898 }

```

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.

```

3899 \newif\if@mainmatter\@mainmattertrue
3900 \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.

```

3901 \keys_define:nn { document-structure / sectioning }{
3902   name .str_set_x:N = \l__document_structure_sect_name_str ,
3903   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
3904   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
3905   num .bool_set:N = \l__document_structure_sect_num_bool ,
3906 }

```

```

3907 \cs_new_protected:Nn \l__document_structure_sect_args:n {
3908   \str_clear:N \l__document_structure_sect_name_str
3909   \str_clear:N \l__document_structure_sect_ref_str
3910   \bool_set_false:N \l__document_structure_sect_clear_bool
3911   \bool_set_false:N \l__document_structure_sect_num_bool
3912   \keys_set:nn { document-structure / sectioning } { #1 }
3913 }
3914 \newcommand\omdoc@sectioning[3][]{
3915   \l__document_structure_sect_args:n {#1}
3916   \let\omdoc@sect@name\l__document_structure_sect_name_str
3917   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
3918   \if@mainmatter% numbering not overridden by frontmatter, etc.
3919     \bool_if:NTF \l__document_structure_sect_num_bool {
3920       \omgroup@num{#2}{#3}
3921     }{
3922       \omgroup@nonum{#2}{#3}
3923     }
3924     \def\current@section@level{\omdoc@sect@name}
3925   \else
3926     \omgroup@nonum{#2}{#3}
3927   \fi
3928 }% 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.

```

3929 \newcommand\omgroup@redefine@addtocontents[1]{%
3930   %\edef\__document_structureimport{#1}%
3931   %\@for\@I:=\__document_structureimport\do{%
3932     %\edef\@path{\csname module@\@I @path\endcsname}%
3933     %\@ifundefined{tf@toc}\relax%
3934     % {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3935   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3936   %\def\addcontentsline##1##2##3{%
3937     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3938   %\else% hyperref.sty not loaded
3939   %\def\addcontentsline##1##2##3{%
3940     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{#1}{##3}}{\thepage}}%
3941   %\fi
3942 }% 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.

```

3943 \int_new:N \l__document_structure_omgroup_level_int
3944 \newenvironment{omgroup}[2][]{% keys, title
3945 {
3946   \l__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.

```

3947 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
3948   \omgroup@redefine@addtocontents{
3949     %\@ifundefined{module@id}\used@modules%
3950     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

3951     }
3952 }

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

3953 \int_incr:N \l_document_structure_omgroup_level_int
3954 \int_incr:N \l_document_structure_section_level_int
3955 \ifcase\l_document_structure_section_level_int
3956   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3957   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3958   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3959   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3960   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3961   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3962   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3963 \fi
3964 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
3965 \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str
3966 }% for customization
3967 {}

```

and finally, we localize the sections

```

3968 \newcommand\omdoc@part@kw{Part}
3969 \newcommand\omdoc@chapter@kw{Chapter}
3970 \newcommand\omdoc@section@kw{Section}
3971 \newcommand\omdoc@subsection@kw{Subsection}
3972 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3973 \newcommand\omdoc@paragraph@kw{paragraph}
3974 \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`

```

3975 \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).

```

3976 \cs_if_exist:NTF\frontmatter{
3977   \let\__document_structure_orig_frontmatter\frontmatter
3978   \let\frontmatter\relax
3979 }{
3980   \tl_set:Nn\__document_structure_orig_frontmatter{
3981     \clearpage
3982     \@mainmatterfalse
3983     \pagenumbering{roman}
3984   }
3985 }
3986 \cs_if_exist:NTF\backmatter{

```

```

3987 \let\__document_structure_orig_backmatter\backmatter
3988 \let\backmatter\relax
3989 }{
3990 \tl_set:Nn\__document_structure_orig_backmatter{
3991 \clearpage
3992 \@mainmatterfalse
3993 \pagenumbering{roman}
3994 }
3995 }

```

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.

```

3996 \newenvironment{frontmatter}{
3997 \__document_structure_orig_frontmatter
3998 }{
3999 \cs_if_exist:NTF\mainmatter{
4000 \mainmatter
4001 }{
4002 \clearpage
4003 \@mainmattertrue
4004 \pagenumbering{arabic}
4005 }
4006 }

```

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

```

4007 \newenvironment{backmatter}{
4008 \__document_structure_orig_backmatter
4009 }{
4010 \cs_if_exist:NTF\mainmatter{
4011 \mainmatter
4012 }{
4013 \clearpage
4014 \@mainmattertrue
4015 \pagenumbering{arabic}
4016 }
4017 }

```

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

```

4018 \@mainmattertrue\pagenumbering{arabic}

```

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

```

4019 \newcommand\afterprematurestop{}
4020 \def\prematurestop@endomgroup{
4021 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
4022 \end{omgroup}
4023 \int_decr:N \l_document_structure_omgroup_level_int
4024 \prematurestop@endomgroup
4025 }
4026 }
4027 \providecommand\prematurestop{

```

```

4028 \message{Stopping sTeX processing prematurely}
4029 \prematurestop@endomgroup
4030 \afterprematurestop
4031 \end{document}
4032 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

4033 \RequirePackage{etoolbox}
4034 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

4035 \newrobustcmd\useSGvar[1]{%
4036   \@ifundefined{sTeX@Gvar@#1}
4037   {\PackageError{omdoc}
4038     {The sTeX Global variable #1 is undefined}
4039     {set it with \protect\setSGvar}}
4040   \@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.

```

4041 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
4042   \@ifundefined{sTeX@Gvar@#1}
4043   {\PackageError{omdoc}
4044     {The sTeX Global variable #1 is undefined}
4045     {set it with \protect\setSGvar}}
4046   {\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.

```
4047 \*cls)
4048 \@@=mikoslides)
4049 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4050 \RequirePackage{13keys2e,expl-keystr-compatible}
4051
4052 \keys_define:nn{mikoslides / cls}{
4053   class .code:n = {
4054     \PassOptionsToClass{\CurrentOption}{omdoc}
4055     \str_if_eq:nnT{#1}{book}{
4056       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4057     }
4058     \str_if_eq:nnT{#1}{report}{
4059       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4060     }
4061   },
4062   notes .bool_set:N = \c__mikoslides_notes_bool ,
4063   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4064   unknown .code:n = {
4065     \PassOptionsToClass{\CurrentOption}{omdoc}
4066     \PassOptionsToClass{\CurrentOption}{beamer}
4067     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4068   }
4069 }
4070 \ProcessKeysOptions{ mikoslides / cls }
4071 \bool_if:NTF \c__mikoslides_notes_bool {
4072   \PassOptionsToPackage{notes=true}{mikoslides}
4073 }{
4074   \PassOptionsToPackage{notes=false}{mikoslides}
4075 }
4076 \</cls)
```

now we do the same for the mikoslides package.

```

4077 <*package>
4078 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4079 \RequirePackage{l3keys2e,expl-keystr-compat}
4080
4081 \keys_define:nn{mikoslides / pkg}{
4082   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
4083   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4084   notes            .bool_set:N = \c__mikoslides_notes_bool ,
4085   slides           .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4086   sectocframes     .bool_set:N = \c__mikoslides_sectocframes_bool ,
4087   frameimages      .bool_set:N = \c__mikoslides_frameimages_bool ,
4088   fiboxed          .bool_set:N = \c__mikoslides_fiboxed_bool ,
4089   noproblems       .bool_set:N = \c__mikoslides_noproblems_bool,
4090   unknown          .code:n      = {
4091     \PassOptionsToClass{\CurrentOption}{stex}
4092     \PassOptionsToClass{\CurrentOption}{tikzinput}
4093   }
4094 }
4095 \ProcessKeysOptions{ mikoslides / pkg }
4096 \newif\ifnotes
4097 \bool_if:NTF \c__mikoslides_notes_bool {
4098   \notesttrue
4099 }{
4100   \notesfalse
4101 }
4102

```

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

```

4103 \str_if_empty:NTF \c__mikoslides_topsect_str {
4104   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4105 }{
4106   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4107 }
4108 </package>

```

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

```

4109 <*cls>
4110 \bool_if:NTF \c__mikoslides_notes_bool {
4111   \LoadClass{omdoc}
4112 }{
4113   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4114   \newcounter{Item}
4115   \newcounter{paragraph}
4116   \newcounter{subparagraph}
4117   \newcounter{Hfootnote}
4118   \RequirePackage{omdoc}
4119 }

```

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

```

4120 \RequirePackage{mikoslides}
4121 </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.

```

4122 <*package>
4123 \bool_if:NT \c__mikoslides_notes_bool {
4124   \RequirePackage{a4wide}
4125   \RequirePackage{marginnote}
4126   \PassOptionsToPackage{usenames,dvipsnames,svgnames}{xcolor}
4127   \RequirePackage{mdframed}
4128   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4129   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4130 }
4131 \RequirePackage{stex-compatibility}
4132 \RequirePackage{stex-tikzinput}
4133 \RequirePackage{etoolbox}
4134 \RequirePackage{amssymb}
4135 \RequirePackage{amsmath}
4136 \RequirePackage{comment}
4137 \RequirePackage{textcomp}
4138 \RequirePackage{url}
4139 \RequirePackage{graphicx}
4140 \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

```

4141 \bool_if:NT \c__mikoslides_notes_bool {
4142   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4143 }

```

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

```

4144 \newcounter{slide}
4145 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4146 \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.

```

4147 \bool_if:NTF \c__mikoslides_notes_bool {
4148   \renewenvironment{note}{\ignorespaces}{\ignorespaces}{}
4149 }{
4150   \excludecomment{note}
4151 }

```

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.

```
4152 \bool_if:NT \c__mikoslides_notes_bool {
4153   \newlength{\slideframewidth}
4154   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4155 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4156   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4157     \bool_set_true:N #1
4158   }{
4159     \bool_set_false:N #1
4160   }
4161 }
4162 \keys_define:nn{mikoslides / frame}{
4163   label .str_set_x:N = \l__mikoslides_frame_label_str,
4164   allowframebreaks .code:n = {
4165     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4166   },
4167   allowdisplaybreaks .code:n = {
4168     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4169   },
4170   fragile .code:n = {
4171     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4172   },
4173   shrink .code:n = {
4174     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4175   },
4176   squeeze .code:n = {
4177     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4178   },
4179   t .code:n = {
4180     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4181   },
4182 }
4183 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4184   \str_clear:N \l__mikoslides_frame_label_str
4185   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4186   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4187   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4188   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4189   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4190   \bool_set_true:N \l__mikoslides_frame_t_bool
4191   \keys_set:nn { mikoslides / frame }{ #1 }
4192 }
```

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

```
4193 \renewenvironment{frame}[1][]{
4194   \__mikoslides_frame_args:n{#1}
4195   \sffamily
4196   \stepcounter{slide}
4197   \def\@currentlabel{\theslide}
4198   \str_if_empty:NF \l__mikoslides_frame_label_str {
4199     \label{\l__mikoslides_frame_label_str}
```

```
4200 }
```

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

```
4201 \def\itemize@level{outer}
4202 \def\itemize@outer{outer}
4203 \def\itemize@inner{inner}
4204 \renewcommand\newpage{\addtocounter{framenum}{1}}
4205 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4206 \renewenvironment{itemize}{
4207   \ifx\itemize@level\itemize@outer
4208     \def\itemize@label{\$ \rhd \$}
4209   \fi
4210   \ifx\itemize@level\itemize@inner
4211     \def\itemize@label{\$ \scriptstyle \rhd \$}
4212   \fi
4213   \begin{list}
4214     {\itemize@label}
4215     {\setlength{\labelsep}{.3em}
4216      \setlength{\labelwidth}{.5em}
4217      \setlength{\leftmargin}{1.5em}
4218     }
4219   \edef\itemize@level{\itemize@inner}
4220 }{
4221   \end{list}
4222 }
```

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

```
4223 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4224 }{
4225   \medskip\miko@slidelabel\end{mdframed}
4226 }
```

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

`\frametitle`

```
4227 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4228 }
```

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

EdN:18

`\pause` 18

```
4229 \bool_if:NT \c__mikoslides_notes_bool {
4230   \newcommand\pause{}
4231 }
```

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

`nomtext`

```
4232 \bool_if:NTF \c__mikoslides_notes_bool {
4233   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}{
4234 }{
4235   \excludecomment{nomtext}
4236 }
```

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

nomgroup

```
4237 \bool_if:NTF \c__mikoslides_notes_bool {  
4238   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}  
4239 }{  
4240   \excludecomment{nomgroup}  
4241 }
```

ndefinition

```
4242 \bool_if:NTF \c__mikoslides_notes_bool {  
4243   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}  
4244 }{  
4245   \excludecomment{ndefinition}  
4246 }
```

nassertion

```
4247 \bool_if:NTF \c__mikoslides_notes_bool {  
4248   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}  
4249 }{  
4250   \excludecomment{nassertion}  
4251 }
```

nsproof

```
4252 \bool_if:NTF \c__mikoslides_notes_bool {  
4253   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}  
4254 }{  
4255   \excludecomment{nsproof}  
4256 }
```

nexample

```
4257 \bool_if:NTF \c__mikoslides_notes_bool {  
4258   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}  
4259 }{  
4260   \excludecomment{nexample}  
4261 }
```

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

```
4262 \def\inputref@preskip{\smallskip}  
4263 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
4264 \let\orig@inputref\inputref  
4265 \def\inputref{\@ifstar\ninputref\orig@inputref}  
4266 \newcommand\ninputref[2] [] {  
4267   \bool_if:NT \c__mikoslides_notes_bool {  
4268     \orig@inputref[#1]{#2}  
4269   }  
4270 }
```

(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>}`.

```

4271 \newlength{\slidelogoheight}
4272
4273 \bool_if:NTF \c__mikoslides_notes_bool {
4274   \setlength{\slidelogoheight}{.4cm}
4275 }{
4276   \setlength{\slidelogoheight}{1cm}
4277 }
4278 \newsavebox{\slidelogo}
4279 \sbox{\slidelogo}{\TeX}
4280 \newrobustcmd{\setslidelogo}[1]{
4281   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4282 }
```

(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.

```

4283 \def\source{Michael Kohlhase}% customize locally
4284 \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.

```

4285 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4286 \newsavebox{\cclogo}
4287 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4288 \newif\ifcchref\cchreffalse
4289 \AtBeginDocument{
4290   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4291 }
4292 \def\licensing{
4293   \ifcchref
4294     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4295   \else
4296     {\usebox{\cclogo}}
4297   \fi
4298 }
4299 \newrobustcmd{\setlicensing}[2][]{
4300   \def\@url{#1}
4301   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4302   \ifx\@url\@empty
4303     \def\licensing{{\usebox{\cclogo}}}
4304   \else
4305     \def\licensing{
```

```

4306     \ifcchref
4307     \href{#1}{\usebox{\cclogo}}
4308   \else
4309     {\usebox{\cclogo}}
4310   \fi
4311 }
4312 \fi
4313 }

```

(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

```

4314 \newrobustcmd\miko@slidelabel{
4315   \vbox to \slidelogoheight{
4316     \vss\hbox to \slidewidth
4317     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4318   }
4319 }

```

(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.

```

4320 \def\Gin@mhrepos{}
4321 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4322 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4323 \newrobustcmd\frameimage[2][{}]{
4324   \stepcounter{slide}
4325   \bool_if:NT \c__mikoslides_frameimages_bool {
4326     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4327     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4328     \begin{center}
4329       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4330         \fbox{
4331           \ifx\Gin@ewidth\@empty
4332             \ifx\Gin@mhrepos\@empty
4333               \mhgraphics[width=\slidewidth,#1]{#2}
4334             \else
4335               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4336             \fi
4337           \else% Gin@ewidth empty
4338             \ifx\Gin@mhrepos\@empty
4339               \mhgraphics[#1]{#2}
4340             \else
4341               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4342             \fi
4343           \fi% Gin@ewidth empty
4344         }
4345       }{
4346         \ifx\Gin@ewidth\@empty

```

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


```

4347         \ifx\Gin@mhrepos\empty
4348             \mhgraphics[width=\slidewidth,#1]{#2}
4349         \else
4350             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4351         \fi
4352         \ifx\Gin@mhrepos\empty
4353             \mhgraphics[#1]{#2}
4354         \else
4355             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4356         \fi
4357     \fi% Gin@ewidth empty
4358 }
4359 \end{center}
4360 \par\strut\hfill{\footnotesize Slide \arabic{slide}}%
4361 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4362 }
4363 } % 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.

```

4364 \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

```

4365 \AddToHook{begindocument}{
4366     \definecolor{green}{rgb}{0,.5,0}
4367     \definecolor{purple}{cmyk}{.3,1,0,.17}
4368 }

```

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`.

```

4369 % \def\STpresent#1{\textcolor{blue}{#1}}
4370 \def\defemph#1{\textcolor{magenta}{#1}}
4371 \def\symrefemph#1{\textcolor{cyan}{#1}}
4372 \def\compemph#1{\textcolor{blue}{#1}}
4373 \def\titleemph#1{\textcolor{blue}{#1}}
4374 \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.

```

4375 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4376 \def\smalltextwarning{
4377     \pgfuseimage{miko@small@dbend}
4378     \xspace
4379 }
4380 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4381 \newrobustcmd\textwarning{
4382   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4383   \xspace
4384 }
4385 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4386 \newrobustcmd\bigtextwarning{
4387   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4388   \xspace
4389 }

```

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

```

4390 \newrobustcmd\putgraphicsat[3]{
4391   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4392 }
4393 \newrobustcmd\putat[2]{
4394   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4395 }

```

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`.

```

4396 \bool_if:NT \c__mikoslides_sectocframes_bool {
4397   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4398     \newcounter{chapter}\counterwithin*{section}{chapter}
4399   }{
4400     \str_if_eq:VnT \__mikoslidestopsect{chapter}{
4401       \newcounter{chapter}\counterwithin*{section}{chapter}
4402     }
4403   }
4404 }

```

`\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
4405 \def\part@prefix{}
4406 \@ifpackageloaded{omdoc}{}{
4407   \str_case:VnF \__mikoslidestopsect {
4408     {part}{
4409       \int_set:Nn \l_document_structure_section_level_int {0}
4410       \def\thesection{\arabic{chapter}.\arabic{section}}
4411       \def\part@prefix{\arabic{chapter}.}
4412     }
4413     {chapter}{
4414       \int_set:Nn \l_document_structure_section_level_int {1}
4415       \def\thesection{\arabic{chapter}.\arabic{section}}
4416       \def\part@prefix{\arabic{chapter}.}
4417     }
4418   }{
4419     \int_set:Nn \l_document_structure_section_level_int {2}
4420     \def\part@prefix{}

```

```

4421 }
4422 }
4423
4424 \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

```

4425 \renewenvironment{omgroup}[2][]{
4426   \__document_structure_omgroup_args:n { #1 }
4427   \int_incr:N \l_document_structure_omgroup_level_int
4428   \int_incr:N \l_document_structure_section_level_int
4429   \bool_if:NT \c__mikoslides_sectocframes_bool {
4430     \stepcounter{slide}
4431     \begin{frame}[noframenumbering]
4432     \vfill\Large\centering
4433     \red{
4434       \ifcase\l_document_structure_section_level_int\or
4435         \stepcounter{part}
4436         \def\__mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4437         \def\currentsectionlevel{\omdoc@part@kw}
4438       \or
4439         \stepcounter{chapter}
4440         \def\__mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4441         \def\currentsectionlevel{\omdoc@chapter@kw}
4442       \or
4443         \stepcounter{section}
4444         \def\__mikoslideslabel{\part@prefix\arabic{section}}
4445         \def\currentsectionlevel{\omdoc@section@kw}
4446       \or
4447         \stepcounter{subsection}
4448         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4449         \def\currentsectionlevel{\omdoc@subsection@kw}
4450       \or
4451         \stepcounter{subsubsection}
4452         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4453         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4454       \or
4455         \stepcounter{mparagraph}
4456         \def\__mikoslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4457         \def\currentsectionlevel{\omdoc@paragraph@kw}
4458       \fi% end ifcase
4459       \__mikoslideslabel%\sref@label@id\__mikoslideslabel
4460       \quad #2%
4461     }%
4462     \vfill%
4463     \end{frame}%
4464   }
4465   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4466 }{}
4467 }

```

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

```

4468 \def\inserttheorembodyfont{\normalfont}
4469 \bool_if:NF \c__mikoslices_notes_bool {
4470   \defbeamertemplate{theorem begin}{miko}
4471   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4472     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4473     \inserttheorempunctuation\inserttheorembodyfont\space}
4474   \defbeamertemplate{theorem end}{miko}{}
```

and we set it as the default one.

```

4475   \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.

```

4476   \expandafter\def\csname Parent2\endcsname{}
4477 }
4478 \bool_if:NT \c__mikoslices_notes_bool {
4479   \renewenvironment{columns}[1][{}]{%
4480     \par\noindent%
4481     \begin{minipage}%
4482       \slidewidth\centering\leavevmode%
4483   }{%
4484     \end{minipage}\par\noindent%
4485   }%
4486   \newsavebox\columnbox%
4487   \renewenvironment<>{column}[2][{}]{%
4488     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4489   }{%
4490     \end{minipage}\end{lrbox}\usebox\columnbox%
4491   }%
4492 }
4493 \bool_if:NTF \c__mikoslices_noproblems_bool {
4494   \newenvironment{problems}{}{}
4495 }{
4496   \excludecomment{problems}
4497 }
```

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.

```

4498 \gdef\printexcursions{}
4499 \newcommand\excursionref[2]{% label, text
4500   \bool_if:NT \c__mikoslices_notes_bool {
4501     \begin{omtext}[title=Excursion]
4502       #2 \sref[fallback=the appendix]{#1}.
4503     \end{omtext}
4504   }
4505 }
4506 \newcommand\activate@excursion[2][{}]{
4507   \gappto\printexcursions{\inputref[#1]{#2}}
```

```

4508 }
4509 \newcommand\excursion[4][{}]{% repos, label, path, text
4510   \bool_if:NT \c__mikoslides_notes_bool {
4511     \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4512   }
4513 }

```

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

\excursiongroup

```

4514 \keys_define:nn{mikoslides / excursiongroup }{
4515   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4516   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4517   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4518 }
4519 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4520   \tl_clear:N \l__mikoslides_excursion_intro_tl
4521   \str_clear:N \l__mikoslides_excursion_id_str
4522   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4523   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4524 }
4525 \newcommand\excursiongroup[1][{}]{
4526   \__mikoslides_excursion_args:n{ #1 }
4527   \ifdefempty\printexcursions{}% only if there are excursions
4528   {\begin{note}
4529     \begin{omgroup}[#1]{Excursions}%
4530     \ifdefempty\l__mikoslides_excursion_intro_tl{{
4531       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4532         \l__mikoslides_excursion_intro_tl
4533       }
4534     }
4535     \printexcursions%
4536     \end{omgroup}
4537   \end{note}}
4538 }
4539 \end{package}

```

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

Chapter 33

The Implementation

33.1 Package Options

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

```
4540 <*package>
4541 <@@=problems>
4542 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4543 \RequirePackage{l3keys2e,expl-keystr-compatible}
4544
4545 \keys_define:nn { problem / pkg }{
4546   notes      .default:n    = { true },
4547   notes      .bool_set:N   = \c__problems_notes_bool,
4548   gnotes     .default:n    = { true },
4549   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4550   hints      .default:n    = { true },
4551   hints      .bool_set:N   = \c__problems_hints_bool,
4552   solutions  .default:n    = { true },
4553   solutions  .bool_set:N   = \c__problems_solutions_bool,
4554   pts        .default:n    = { true },
4555   pts        .bool_set:N   = \c__problems_pts_bool,
4556   min        .default:n    = { true },
4557   min        .bool_set:N   = \c__problems_min_bool,
4558   boxed      .default:n    = { true },
4559   boxed      .bool_set:N   = \c__problems_boxed_bool,
4560   unknown    .code:n       = {}
4561 }
4562 \def\solutionstrue{
4563   \bool_set_true:N \c__problems_solutions_bool
4564 }
4565 \def\solutionsfalse{
4566   \bool_set_false:N \c__problems_solutions_bool
4567 }
4568
4569 \ProcessKeysOptions{ problem / pkg }
```

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

```

4570 \RequirePackage{stex-compatibility}
4571 \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.

```

4572 \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.

```

4573 \def\prob@problem@kw{Problem}
4574 \def\prob@solution@kw{Solution}
4575 \def\prob@hint@kw{Hint}
4576 \def\prob@note@kw{Note}
4577 \def\prob@gnote@kw{Grading}
4578 \def\prob@pt@kw{pt}
4579 \def\prob@min@kw{min}

```

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

For the other languages, we set up triggers

```

4580 \@ifpackageloaded{babel}{
4581   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4582   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4583     \input{problem-ngerman.ldf}
4584   }
4585   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4586     \input{problem-finnish.ldf}
4587   }
4588   \clist_if_in:NnT \l_tmpa_clist {french}{
4589     \input{problem-french.ldf}
4590   }
4591   \clist_if_in:NnT \l_tmpa_clist {russian}{
4592     \input{problem-russian.ldf}
4593   }
4594 }{}

```

33.2 Problems and Solutions

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

```

4595 \keys_define:nn{ problem / problem }{
4596   id      .str_set:x:N = \l__problems_prob_id_str,
4597   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4598   min     .tl_set:N    = \l__problems_prob_min_tl,
4599   title   .tl_set:N    = \l__problems_prob_title_tl,
4600   refnum  .int_set:N   = \l__problems_prob_refnum_int
4601 }
4602 \cs_new_protected:Nn \__problems_prob_args:n {
4603   \str_clear:N \l__problems_prob_id_str
4604   \tl_clear:N \l__problems_prob_pts_tl
4605   \tl_clear:N \l__problems_prob_min_tl
4606   \tl_clear:N \l__problems_prob_title_tl

```

```

4607 \int_zero_new:N \l__problems_prob_refnum_int
4608 \keys_set:nn { problem / problem }{ #1 }
4609 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4610   \let\l__problems_inclprob_refnum_int\undefined
4611 }
4612 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4613 \newcounter{problem}
4614 \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.

```

4615 \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

```

4616 \newcommand\prob@number{
4617   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4618     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4619   }{
4620     \int_if_exist:NTF \l__problems_prob_refnum_int {
4621       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4622     }{
4623       \prob@label\theproblem
4624     }
4625   }
4626 }

```

(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.

```

4627 \newcommand\prob@title[3]{%
4628   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4629     #2 \l__problems_inclprob_title_tl #3
4630   }{
4631     \tl_if_exist:NTF \l__problems_prob_title_tl {
4632       #2 \l__problems_prob_title_tl #3
4633     }{
4634       #1
4635     }
4636   }
4637 }

```

(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.

```

4638 \def\prob@heading{
4639   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4640   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4641 }

```

(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`

```

4642 \newenvironment{problem}[1][1]{
4643   \_problems_prob_args:n{#1}%\sref@target%
4644   \@in@omtexttrue% we are in a statement (for inline definitions)
4645   \stepcounter{problem}\record@problem
4646   \def\current@section@level{\prob@problem@kw}
4647   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4648 }%
4649   {\smallskip}
4650   \bool_if:NT \c__problems_boxed_bool {
4651     \surroundwithmdframed{problem}
4652   }

```

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

```

4653 \def\record@problem{
4654   \protected@write\@auxout{}
4655   {
4656     \string\@problem{\prob@number}
4657     {
4658       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4659         \l__problems_inclprob_pts_tl
4660       }{
4661         \l__problems_prob_pts_tl
4662       }
4663     }%
4664     {
4665       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4666         \l__problems_inclprob_min_tl
4667       }{
4668         \l__problems_prob_min_tl
4669       }
4670     }
4671   }
4672 }

```

(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).

```

4673 \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.

```

4674 \keys_define:nn { problem / solution }{
4675   id          .str_set_x:N = \l__problems_solution_id_str ,
4676   for         .tl_set:N    = \l__problems_solution_for_tl ,
4677   height      .dim_set:N   = \l__problems_solution_height_dim ,
4678   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4679   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4680   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4681 }
4682 \cs_new_protected:Nn \__problems_solution_args:n {
4683   \str_clear:N \l__problems_solution_id_str
4684   \tl_clear:N \l__problems_solution_for_tl
4685   \tl_clear:N \l__problems_solution_srccite_tl
4686   \clist_clear:N \l__problems_solution_creators_clist
4687   \clist_clear:N \l__problems_solution_contributors_clist
4688   \dim_zero:N \l__problems_solution_height_dim
4689   \keys_set:nn { problem / solution }{ #1 }
4690 }

```

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

```

4691 \newcommand\@startsolution[1][ ]{
4692   \__problems_solution_args:n { #1 }
4693   \@in@omtexttrue% we are in a statement.
4694   \bool_if:NF \c__problems_boxed_bool { \hrule }
4695   \smallskip\noindent
4696   {\textbf\prob@solution@kw : \enspace}
4697   \begin{small}
4698   \def\current@section@level{\prob@solution@kw}
4699   \ignorespacesandpars
4700 }

```

\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.

```

4701 \newcommand\startsolutions{
4702   \specialcomment{solution}{\@startsolution}{
4703     \bool_if:NF \c__problems_boxed_bool {
4704       \hrule\medskip
4705     }
4706     \end{small}%
4707   }
4708   \bool_if:NT \c__problems_boxed_bool {
4709     \surroundwithmdframed{solution}
4710   }
4711 }

```

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

\stopsolutions

```

4712 \newcommand\stopsolutions{\excludecomment{solution}}

```

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

so it only remains to start/stop solutions depending on what option was specified.

```

4713 \bool_if:NTF \c__problems_solutions_bool {
4714   \startsolutions
4715 }{
4716   \stopsolutions
4717 }

```

exnote

```

4718 \bool_if:NTF \c__problems_notes_bool {
4719   \newenvironment{exnote}[1][]{
4720     \par\smallskip\hrule\smallskip
4721     \noindent\textbf{\prob@note@kw : }\small
4722   }{
4723     \smallskip\hrule
4724   }
4725 }{
4726   \excludecomment{exnote}
4727 }

```

hint

```

4728 \bool_if:NTF \c__problems_notes_bool {
4729   \newenvironment{hint}[1][]{
4730     \par\smallskip\hrule\smallskip
4731     \noindent\textbf{\prob@hint@kw :~ }\small
4732   }{
4733     \smallskip\hrule
4734   }
4735   \newenvironment{exhint}[1][]{
4736     \par\smallskip\hrule\smallskip
4737     \noindent\textbf{\prob@hint@kw :~ }\small
4738   }{
4739     \smallskip\hrule
4740   }
4741 }{
4742   \excludecomment{hint}
4743   \excludecomment{exhint}
4744 }

```

gnote

```

4745 \bool_if:NTF \c__problems_notes_bool {
4746   \newenvironment{gnote}[1][]{
4747     \par\smallskip\hrule\smallskip
4748     \noindent\textbf{\prob@gnote@kw : }\small
4749   }{
4750     \smallskip\hrule
4751   }
4752 }{
4753   \excludecomment{gnote}
4754 }

```

33.3 Multiple Choice Blocks

```

4755 \newenvironment{mcb}{
4756   \begin{enumerate}
4757 }{
4758   \end{enumerate}
4759 }

```

we define the keys for the mcc macro

```

4760 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4761   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4762     \bool_set_true:N #1
4763   }{
4764     \bool_set_false:N #1
4765   }
4766 }
4767 \keys_define:nn { problem / mcc }{
4768   id          .str_set:x:N = \l__problems_mcc_id_str ,
4769   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4770   T           .default:n   = { true } ,
4771   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4772   F           .default:n   = { true } ,
4773   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4774   Ttext       .code:n      = {
4775     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4776   } ,
4777   Ftext       .code:n      = {
4778     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4779   }
4780 }
4781 \cs_new_protected:Nn \l__problems_mcc_args:n {
4782   \str_clear:N \l__problems_mcc_id_str
4783   \tl_clear:N \l__problems_mcc_feedback_tl
4784   \bool_set_true:N \l__problems_mcc_t_bool
4785   \bool_set_true:N \l__problems_mcc_f_bool
4786   \bool_set_true:N \l__problems_mcc_Ttext_bool
4787   \bool_set_false:N \l__problems_mcc_Ftext_bool
4788   \keys_set:nn { problem / mcc }{ #1 }
4789 }

```

\mcc

```

4790 \newcommand\mcc[2][]{
4791   \l__problems_mcc_args:n{ #1 }
4792   \item #2
4793   \bool_if:NT \c__problems_solutions_bool {
4794     \\\
4795     \bool_if:NT \l__problems_mcc_t_bool {
4796       % TODO!
4797       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
4798     }
4799     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

4800      % TODO!
4801      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4802    }
4803    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4804      !
4805    }{
4806      \l__problems_mcc_feedback_tl
4807    }
4808  }
4809 } %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.

```

4810
4811 \keys_define:nn{ problem / inclproblem }{
4812   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4813   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4814   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4815   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4816   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4817   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4818 }
4819
4820 \cs_new_protected:Nn \l__problems_inclprob_args:n {
4821   % \str_clear:N \l__problems_prob_id_str
4822   \tl_clear:N \l__problems_inclprob_pts_tl
4823   \tl_clear:N \l__problems_inclprob_min_tl
4824   \tl_clear:N \l__problems_inclprob_title_tl
4825   \int_zero_new:N \l__problems_inclprob_refnum_int
4826   \str_clear:N \l__problems_inclprob_mhrepos_str
4827   \keys_set:nn { problem / inclproblem }{ #1 }
4828   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4829     \let\l__problems_inclprob_pts_tl\undefined
4830   }
4831   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4832     \let\l__problems_inclprob_min_tl\undefined
4833   }
4834   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4835     \let\l__problems_inclprob_title_tl\undefined
4836   }
4837   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4838     \let\l__problems_inclprob_refnum_int\undefined
4839   }
4840 }
4841
4842 \cs_new_protected:Nn \l__problems_inclprob_clear: {
4843   % \str_clear:N \l__problems_prob_id_str
4844   \let\l__problems_inclprob_pts_tl\undefined
4845   \let\l__problems_inclprob_min_tl\undefined

```

```

4845 \let\l__problems_inclprob_title_tl\undefined
4846 \let\l__problems_inclprob_refnum_int\undefined
4847 \let\l__problems_inclprob_mhrepos_str\undefined
4848 }
4849
4850 \newcommand\includeproblem[2][]{
4851   \__problems_inclprob_args:n{ #1 }
4852   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4853     \input{#2}
4854   }{
4855     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4856       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4857     }
4858   }
4859   \__problems_inclprob_clear:
4860 }

```

(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.

```

4861 \AddToHook{enddocument}{
4862   \bool_if:NT \c__problems_pts_bool {
4863     \message{Total:~\arabic{pts}~points}
4864   }
4865   \bool_if:NT \c__problems_min_bool {
4866     \message{Total:~\arabic{min}~minutes}
4867   }
4868 }

```

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

```

4869 \def\pts#1{
4870   \bool_if:NT \c__problems_pts_bool {
4871     \marginpar{#1~\prob@pt@kw}
4872   }
4873 }
4874 \def\min#1{
4875   \bool_if:NT \c__problems_min_bool {
4876     \marginpar{#1~\prob@min@kw}
4877   }
4878 }

```

\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.

```

4879 \newcounter{pts}
4880 \def\show@pts{
4881   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4882     \bool_if:NT \c__problems_pts_bool {
4883       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4884       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

4885     }
4886   }{
4887     \tl_if_exist:NT \l__problems_prob_pts_tl {
4888       \bool_if:NT \c__problems_pts_bool {
4889         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
4890         \addtocounter{pts}{\l__problems_prob_pts_tl}
4891       }
4892     }
4893   }
4894 }

```

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

\show@min

```

4895 \newcounter{min}
4896 \def\show@min{
4897   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4898     \bool_if:NT \c__problems_min_bool {
4899       \marginpar{\l__problems_inclprob_pts_tl;min}
4900       \addtocounter{min}{\l__problems_inclprob_min_tl}
4901     }
4902   }{
4903     \tl_if_exist:NT \l__problems_prob_min_tl {
4904       \bool_if:NT \c__problems_min_bool {
4905         \marginpar{\l__problems_prob_min_tl;min}
4906         \addtocounter{min}{\l__problems_prob_min_tl}
4907       }
4908     }
4909   }
4910 }
4911 \</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.

```
4912 <@@=hwexam>
4913 <*cls>
4914 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4915 \RequirePackage{l3keys2e,expl-keystr-compatible}
4916 \DeclareOption*{
4917   \PassOptionsToClass{\CurrentOption}{omdoc}
4918   \PassOptionsToPackage{\CurrentOption}{stex}
4919   \PassOptionsToPackage{\CurrentOption}{hwexam}
4920   \PassOptionsToPackage{\CurrentOption}{tikzinput}
4921 }
4922 \ProcessOptions
```

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

```
4923 \LoadClass{omdoc}
4924 \RequirePackage{stex}
4925 \RequirePackage{hwexam}
4926 \RequirePackage{tikzinput}
4927 \RequirePackage{graphicx}
4928 \RequirePackage{a4wide}
4929 \RequirePackage{amssymb}
4930 \RequirePackage{amstext}
4931 \RequirePackage{amsmath}
```

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


```

4932 \newcommand\assig@default@type{\hwexam@assignment@kw}
4933 \def\document@hwexamtype{\assig@default@type}
4934 <@@=document_structure>
4935 \keys_define:nn { document-structure / document }{
4936 id .str_set_x:N = \c_document_structure_document_id_str,
4937 hwexamtype .tl_set:N = \document@hwexamtype
4938 }
4939 <@@=hwexam>
4940 </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.

```
4941 \*package>
4942 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4943 \RequirePackage{l3keys2e,expl-keystr-compat}
4944
4945 \newif\iftest\testfalse
4946 \DeclareOption{test}{\testtrue}
4947 \newif\ifmultiple\multiplefalse
4948 \DeclareOption{multiple}{\multipletrue}
4949 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
4950 \ProcessOptions
```

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

```
4951 \RequirePackage{keyval}[1997/11/10]
4952 \RequirePackage{problem}
```

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

```
4953 \newcommand\hwexam@assignment@kw{Assignment}
4954 \newcommand\hwexam@given@kw{Given}
4955 \newcommand\hwexam@due@kw{Due}
4956 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
4957   space}%
4958 \newcommand\correction@probs@kw{prob.}%
4959 \newcommand\correction@pts@kw{total}%
4960 \newcommand\correction@reached@kw{reached}%
4961 \newcommand\correction@sum@kw{Sum}%
4962 \newcommand\correction@grade@kw{grade}%
4963 \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

```

4964 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
4965
4966 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4967 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4968   \input{hwexam-ngerman.ldf}
4969 }
4970 \clist_if_in:NnT \l_tmpa_clist {finnish}{
4971   \input{hwexam-finnish.ldf}
4972 }
4973 \clist_if_in:NnT \l_tmpa_clist {french}{
4974   \input{hwexam-french.ldf}
4975 }
4976 \clist_if_in:NnT \l_tmpa_clist {russian}{
4977   \input{hwexam-russian.ldf}
4978 }

```

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.

```

4979 \newcounter{assignment}
4980 \numberproblemsin{assignment}
4981 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

4982 \keys_define:nn { hwexam / assignment } {
4983   id .str_set:N = \l__hwexam_assign_id_str,
4984   number .int_set:N = \l__hwexam_assign_number_int,
4985   title .tl_set:N = \l__hwexam_assign_title_tl,
4986   type .tl_set:N = \l__hwexam_assign_type_tl,
4987   given .tl_set:N = \l__hwexam_assign_given_tl,
4988   due .tl_set:N = \l__hwexam_assign_due_tl,
4989   loadmodules .code:n = {
4990     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
4991   }
4992 }
4993 \cs_new_protected:Nn \__hwexam_assignment_args:n {
4994   \str_clear:N \l__hwexam_assign_id_str
4995   \int_set:Nn \l__hwexam_assign_number_int {-1}
4996   \tl_clear:N \l__hwexam_assign_title_tl
4997   \tl_clear:N \l__hwexam_assign_type_tl
4998   \tl_clear:N \l__hwexam_assign_given_tl
4999   \tl_clear:N \l__hwexam_assign_due_tl
5000   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
5001   \keys_set:nn { hwexam / assignment }{ #1 }
5002 }

```

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.

```

5003 \newcommand\given@due[2]{
5004 \bool_lazy_all:nF {
5005 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5006 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5007 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5008 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5009 }{ #1 }
5010
5011 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
5012 \tl_if_empty:NF \l__hwexam_assign_given_tl {
5013 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
5014 }
5015 }{
5016 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
5017 }
5018
5019 \bool_lazy_or:nnF {
5020 \bool_lazy_and_p:nn {
5021 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5022 }{
5023 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5024 }
5025 }{
5026 \bool_lazy_and_p:nn {
5027 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
5028 }{
5029 \tl_if_empty_p:V \l__hwexam_assign_due_tl
5030 }
5031 }{ ,~ }
5032
5033 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
5034 \tl_if_empty:NF \l__hwexam_assign_due_tl {
5035 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
5036 }
5037 }{
5038 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
5039 }
5040
5041 \bool_lazy_all:nF {
5042 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
5043 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
5044 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
5045 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5046 }{ #2 }
5047 }

```

`\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.

```

5048 \newcommand\assignment@title[3]{

```

```

5049 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5050 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5051 #1
5052 }{
5053 #2\l__hwexam_assign_title_tl#3
5054 }
5055 }{
5056 #2\l__hwexam_inclassassign_title_tl#3
5057 }
5058 }

```

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

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

```

5059 \newcommand\assignment@number{
5060 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5061 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5062 \int_use:N \l__hwexam_assign_number_int
5063 }
5064 }{
5065 \int_use:N \l__hwexam_inclassassign_number_int
5066 }
5067 }

```

(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.

```

5068 \newenvironment{assignment}[1][]{
5069 \__hwexam_assignment_args:n { #1 }
5070 %\sref@target
5071 \let\__hwexamnum\l__hwexam_assign_number_int
5072 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5073 \stepcounter{assignment}
5074 }{
5075 \setcounter{assignment}{\int_use:N\__hwexamnum}
5076 }
5077 \setcounter{problem}{0}
5078 \def\current@section@level{\document@hwexamtype}
5079 %\sref@label@id{\document@hwexamtype \thesection}
5080 \begin{@assignment}
5081 }{
5082 \end{@assignment}
5083 }

```

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

```

5084 \def\__hwexasstitle{
5085 \protect\document@hwexamtype~\arabic{assignment}
5086 \assignment@title{}\;{} \; -- \given@due{}\}
5087 }

```

```

5088 \ifmultiple
5089 \newenvironment{@assignment}{
5090 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5091 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5092 }{
5093 \begin{omgroup}{\__hwexasstitle}
5094 }
5095 }{
5096 \end{omgroup}
5097 }

```

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

```

5098 \else
5099 \newenvironment{@assignment}{
5100 \begin{center}\bf
5101 \Large\@title\strut\
5102 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:\;}{\}
5103 \large\given@due{--\;}{\;}{--}
5104 \end{center}
5105 }{}
5106 \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.

```

5107 \keys_define:nn { hwexam / inclassignment } {
5108 %id .str_set_x:N = \l__hwexam_assign_id_str,
5109 number .int_set:N = \l__hwexam_inclassign_number_int,
5110 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5111 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5112 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5113 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5114 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5115 }
5116 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5117 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5118 \tl_clear:N \l__hwexam_inclassign_title_tl
5119 \tl_clear:N \l__hwexam_inclassign_type_tl
5120 \tl_clear:N \l__hwexam_inclassign_given_tl
5121 \tl_clear:N \l__hwexam_inclassign_due_tl
5122 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5123 \keys_set:nn { hwexam / inclassignment }{ #1 }
5124 }
5125 \__hwexam_inclassignment_args:n {}
5126
5127 \newcommand\inputassignment[2][ ]{
5128 \__hwexam_inclassignment_args:n { #1 }
5129 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5130 \input{#2}
5131 }{
5132 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5133 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5134 }
5135 }
5136 \__hwexam_inclasssign_assignment_args:n {}
5137 }
5138 \newcommand\includeassignment[2][ ]{
5139 \newpage
5140 \inputassignment[#1]{#2}
5141 }

```

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

35.4 Typesetting Exams

\quizheading

```

5142 \ExplSyntaxOff
5143 \newcommand\quizheading[1]{%
5144 \def\@tas{#1}%
5145 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5146 \ifx\@tas\empty\else%
5147 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5148 \fi%
5149 }
5150 \ExplSyntaxOn

```

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

\testheading

```

5151 \keys_define:nn { hwexam / testheading } {
5152 min .tl_set:N = \l__hwexam_testheading_min_tl,
5153 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5154 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5155 }
5156 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5157 \tl_clear:N \l__hwexam_testheading_min_tl
5158 \tl_clear:N \l__hwexam_testheading_duration_tl
5159 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5160 \keys_set:nn { hwexam / testheading }{ #1 }
5161 }
5162 \newenvironment{testheading}[1][ ]{
5163 \__hwexam_testheading_args:n{ #1 }
5164 \noindent\large{Name:~\hfill
5165 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5166 \begin{center}
5167 \Large\textbf{\@title}\[1ex]
5168 \large\@date\[3ex]
5169 \end{center}
5170 \textbf{You~have~
5171 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5172 \l__hwexam_testheading_min_tl~minutes
5173 }{
5174 \l__hwexam_testheading_duration_tl
5175 }~

```

```

5176 (sharp)~for~the~test
5177 };\
5178 Write~the~solutions~to~the~sheet.
5179 \par\noindent
5180 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5181 \advance\check@time by -\theassignment@totalmin
5182 The~estimated~time~for~solving~this~exam~is~
5183 {\theassignment@totalmin}~minutes,~
5184 leaving~you~{\the\check@time}~minutes~for~revising~
5185 your~exam.
5186
5187 \par\noindent
5188 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5189 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5190 You~can~reach~{\theassignment@totalpts}~points~if~you~
5191 solve~all~problems.~You~will~only~need~
5192 {\l__hwexam_testheading_reqpts_tl}~points~for~a~perfect~score,~
5193 i.e.~\ {\the\bonus@pts}~points~are~bonus~points.
5194 \vfill
5195 \begin{center}
5196 {
5197 \Large\em You~have~ample~time,~so~take~it~slow~
5198 and~avoid~rushing~to~mistakes!\}[2ex]
5199 Different~problems~test~different~skills~and~
5200 knowledge,~so~do~not~get~stuck~on~one~problem.
5201 }
5202 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5203 \end{center}
5204 }{
5205 \newpage
5206 }

```

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

\testspace

```

5207 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5208 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5209 \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.

```

5210 <@=problems>
5211 \renewcommand\@problem[3]{
5212 \stepcounter{assignment@probs}
5213 \def\__problemspts{#2}

```



```

5214 \ifx\__problemspts\@empty\else
5215 \addtocounter{assignment@totalpts}{#2}
5216 \fi
5217 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5218 \xdef\correction@probs{\correction@probs & #1}%
5219 \xdef\correction@pts{\correction@pts & #2}
5220 \xdef\correction@reached{\correction@reached & }
5221 }
5222 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5223 \newcounter{assignment@probs}
5224 \newcounter{assignment@totalpts}
5225 \newcounter{assignment@totalmin}
5226 \def\correction@probs{\correction@probs@kw}%
5227 \def\correction@pts{\correction@pts@kw}%
5228 \def\correction@reached{\correction@reached@kw}%
5229 \def\after@correction@table{}%
5230 \stepcounter{assignment@probs}
5231 \newcommand\correction@table{
5232 \resizebox{\textwidth}{!}{%
5233 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5234 &\multicolumn{\theassignment@probs}{c|}|%|
5235 {\footnotesize\correction@forgrading@kw} &\\ \hline
5236 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5237 \correction@pts & \theassignment@totalpts & \\ \hline
5238 \correction@reached & & \[.7cm]\hline
5239 \end{tabular}}
5240 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5241 \end{package}

```

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

35.5 Leftovers

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

here we define the logos that characterize the assignment

```

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

```

```

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

```