

The \TeX 3 Package *

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

2021-12-20

Abstract

TODO

*Version 3.0 (last revised 2021-12-20)

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	125
25.1	The feature environment	125
25.2	Features	127
26	STEX-Statements Implementation	132
26.1	Definitions	133
26.2	Assertions	134
26.3	Examples	136
26.4	OMText	137
27	The Implementation	138
27.1	Package Options	138
27.2	Proofs	138
27.3	Justifications	144
28	STEX-Others Implementation	146
29	STEX-Metatheory Implementation	147
30	Tikzinput Implementation	150
31	document-structure.sty Implementation	152
31.1	The OMDoc Class	152
31.2	Class Options	152
31.3	Beefing up the document environment	153
31.4	Implementation: OMDoc Package	153
31.5	Package Options	153
31.6	Document Structure	155
31.7	Front and Backmatter	158
31.8	Global Variables	160
32	MiKoSlides – Implementation	161
32.1	Class and Package Options	161
32.2	Notes and Slides	163
32.3	Header and Footer Lines	167
32.4	Frame Images	168
32.5	Colors and Highlighting	169
32.6	Sectioning	170
32.7	Excursions	172

33 The Implementation	174
33.1 Package Options	174
33.2 Problems and Solutions	175
33.3 Multiple Choice Blocks	180
33.4 Including Problems	181
33.5 Reporting Metadata	182
34 Implementation: The hwexam Class	184
34.1 Class Options	184
35 Implementation: The hwexam Package	186
35.1 Package Options	186
35.2 Assignments	187
35.3 Including Assignments	190
35.4 Typesetting Exams	191
35.5 Leftovers	193

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

ab

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

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


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

Example 2

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

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

.

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

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

Example 3

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

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

.

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

Example 4

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

Multiplying again by b yields...

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

Example 5

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

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

¹EdNOTE: TODO

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

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

Example 6

```
\symdef[ args=2,op={+}]{add}{#1 \comp+ #2}
The operator  $\mathbin{\textcolor{teal}{+}}$  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 _x \int _y \int _z f dx dy dz`?

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

Precedences

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

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

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

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

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

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

For example:

Example 10

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

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

1.1.2 Archives and Imports

Namespaces

Ideally, \TeX would use arbitrary URIs for modules, with no forced relationships between the *logical* namespace of a module and the *physical* location of the file declaring the module – like MMT does things.

Unfortunately, \TeX only provides very restricted access to the file system, so we are forced to generate namespaces systematically in such a way that they reflect the physical location of the associated files, so that \TeX can resolve them accordingly. Largely, users need not concern themselves with namespaces at all, but for completeness sake, we describe how they are constructed:

- If `\begin{module}{Foo}` occurs in a file `/path/to/file/Foo[.<lang>].tex` which does not belong to an archive, the namespace is `file://path/to/file`.
- If the same statement occurs in a file `/path/to/file/bar[.<lang>].tex`, the namespace is `file://path/to/file/bar`.

In other words: outside of archives, the namespace corresponds to the file URI with the filename dropped iff it is equal to the module name, and ignoring the (optional) language suffix¹.

If the current file is in an archive, the procedure is the same except that the initial segment of the file path up to the archive's `source`-folder is replaced by the archive's namespace URI.

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 2

sTeX-Basics

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

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

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

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

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

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

image ($\langle\text{boolean}\rangle$) 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 {$\langle\log\text{-}prefix\rangle$} {$\langle\text{message}\rangle$}</code>
-----------------------------	---

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

<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 `<content>` 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> <code><content></code> <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 `<cs>` throw an error, indicating that it is only allowed in the context of `<environments>`.

`\stex_reactivate_macro:N<cs>` reactivates it again, i.e. this happens ideally in the `<begin>`-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> *	
<code>\stex_path_if_absolute:NTF</code> *	

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

.

5.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

.

Test 5

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

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

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

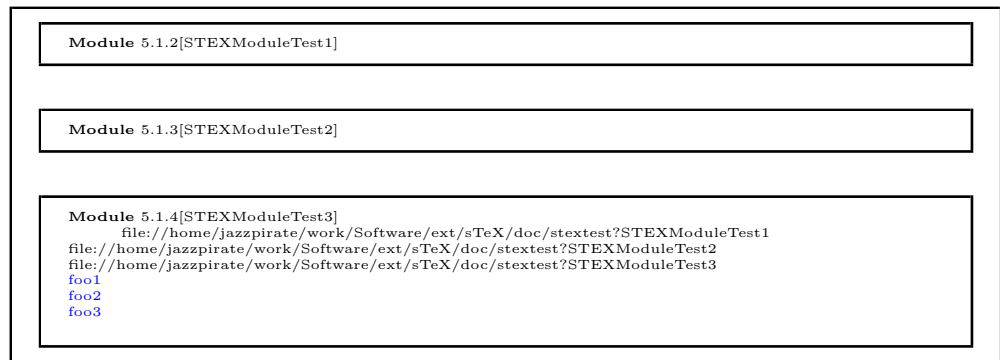
Attempts to find a module whose URI ends with `⟨fragment⟩` in the current scope and passes the full URI on to `\stex_invoke_module:n`.

`\stex_invoke_module:n`

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

Test 6

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

`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

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

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

Test 7

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

6.1.2 Imports and Inheritance

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

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

Test 8

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

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

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

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

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

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

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

Test 9

```

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

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

```

Module 6.1.4[UseTest1]

Module 6.1.5[UseTest2]
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?collec`,
`http://mathhub.info/sTeX?Metatheory?fromto`, `http://mathhub.info/sTeX?Metatheory?apply`, `http://mathhub.info/sTeX?Metatheory?collec`,
`http://mathhub.info/sTeX?Metatheory?seqtype`, `http://mathhub.info/sTeX?Metatheory?sequence-index`, `http://mathhub.info/sTeX?Metatheory?mathematical-structure`,
`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/> <code>\infprec</code> <code>\neginfprec</code> <hr/>	Maximal and minimal notation precedences.
<hr/> <code>\dobrackets</code> <hr/>	<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/> <code>\withbrackets</code> <hr/>	<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]
 $\langle x20x20a^b{}_c \rangle$ and $\langle x20x20a^b{}_c \rangle$.

Test 15

```

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

```

Module 8.1.2[MathTest2]
 $\langle x20x20a|[b,c,d,e,f]^g \rangle$ and $\langle x20x20a|[b,c]^g \rangle$ and $\langle x20x20a|[b]^c \rangle$
 $a+(b\cdot c)$ 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`

`\comp{<args>}`

`\compemph`

`\compemph@uri`

`\defemph`

`\defemph@uri`

`\symrefemph`

`\symrefemph@uri`

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

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

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

`\STEXinvisible`

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

`\ellipses`

TODO

Chapter 9

STEX-Structural Features

Code related to structural features

9.1 Macros and Environments

9.1.1 Structures

mathstructure TODO

Test 17

```

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

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

\present\magma

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

Test: $\mM{op}ab$

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

```

Module 9.1.1[StructureTest1]

a**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

sTeX-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 .

11.2.5 Proof End Markers

Traditionally, the end of a mathematical proof is marked with a little box at the end of the last line of the proof (if there is space and on the end of the next line if there isn't), like so:

`\sproofend`

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

`\sProofEndSymbol`

Some of the proof structuring macros above will insert proof end symbols for sub-proofs, in most cases, this is desirable to make the proof structure explicit, but sometimes this wastes space (especially, if a proof ends in a case analysis which will supply its own proof end marker). To suppress it locally, just set `proofend={}` in them or use `\sProofEndSymbol{}`.

11.2.6 Configuration of the Presentation

Finally, we provide configuration hooks in Figure 1 for the keywords in proofs. These are mainly intended for package authors building on `statements`, e.g. for multi-language support.⁵ The proof step labels can be customized via the `\pstlabelstyle` macro:

EdN:5

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

Figure 1: Configuration Hooks for Semantic Proof Markup

`\pstlabelstyle`

`\pstlabelstyle{<style>}` sets the style; see Figure 2 for an overview of styles. Package writers can add additional styles by adding a macro `\pst@make@label@<style>` that takes two arguments: a comma-separated list of ordinals that make up the prefix and the current ordinal. Note that comma-separated lists can be conveniently iterated over by the L^AT_EX `\@for...:=... \do{...}` macro; see Figure 2 for examples.

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

Figure 2: Configuration Proof Step Label Styles

11.3 Limitations

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

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

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

Chapter 12

sTeX-Metatheory

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

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

12.1 Symbols

Part III
Extensions

Chapter 13

Tikzinput

13.1 Macros and Environments

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

Chapter 14

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

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

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

14.1 Introduction

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

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

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

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

14.2 The User Interface

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

14.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

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

14.2.2 Document Structure

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

`omgroup` The structure of the document is given by the `omgroup` environment just like in OMDoc. 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`[$\langle URL \rangle$]{ $\langle label \rangle$ }, which expands to the previously stored content. If the `\STRlabel` macro was in a different file, then we can give a URL $\langle URL \rangle$ that lets \LaTeX ML generate the correct reference.

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

14.2.5 Global Variables

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

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

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

14.2.6 Colors

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

14.3 Limitations

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

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

Chapter 15

Slides and Course Notes

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

15.1 Introduction

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

15.2 The User Interface

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

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

15.2.1 Package Options

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

- | | |
|---------------------------|--|
| <code>slides</code> | <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, the `\printexcursions` macro (usually in the
 appendix) will collect up all excursions that are specified in the main text.

 Sometimes, we want to reference – in an excursion – part of another. We can use
`\excursionref` `\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{omgroup}[id=<id>]{Excursions}
  \inputref{<path>}
  \printexcursions
\end{omgroup}
```

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.

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: 97 \ifcsname if@latexml\endcsname\else
\latexml_if:TF 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_EX_ML, 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_EX_ML-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:nnx{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:NnN \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 Warnings and error messages
641
642 \iow_new:N \c__stex_refs_refs_iow
643 \AddToHook{begindocument}{
644   \iow_open:Nn \c__stex_refs_refs_iow {\jobname.sref}
645 }
646 \AddToHook{enddocument}{
647   \iow_close:N \c__stex_refs_refs_iow
648 }
649
650 \str_set:Nn \g__stex_refs_title_tl {Unnamed~Document}
651
652 \NewDocumentCommand \STEXreftitle { m } {
653   \tl_gset:Nx \g__stex_refs_title_tl { #1 }
654 }
655
```

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 }
663
```

```

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 \cs_new_protected:Nn \stex_ref_new_doc_target:n {
739   \stex_get_document_uri:
740   \str_set:Nx \l_tmpa_str { #1 }
741   \str_if_empty:NT \l_tmpa_str {
742     \int_zero:N \l_tmpa_int
743     \bool_set_true:N \l_tmpa_bool
744     \bool_while_do:Nn \l_tmpa_bool {
745       \cs_if_exist:cTF {
746         sref_\l_stex_current_docns_str\c_hash_str REF\int_use:N \l_tmpa_int _type
747       }{
748         \int_incr:N \l_tmpa_int
749       }{}
750       \str_set:Nx \l_tmpa_str { REF\int_use:N \l_tmpa_int }
751       \bool_set_false:N \l_tmpa_bool
752     }
753   }
754 }
755 \str_set:Nx \l_tmpa_str {
756   \l_stex_current_docns_str\c_hash_str\l_tmpa_str
757 }
758 \seq_gput_right:No \g__stex_refs_all_refs_seq \l_tmpa_str
759 \stex_if_smsmode:TF {
760   \stex_get_document_url:
761   \str_gset_eq:cN {sref_url_\l_tmpa_str _str}\l_stex_current_docurl_str
762   \str_gset_eq:cN {sref_\l_tmpa_str _type}\c__stex_refs_url_str
763 }{
764   \iow_now:Nx \c__stex_refs_refs_iow { \l_tmpa_str~::~\expandafter{\@currentlabel~in~\exp_a
765   \exp_after:wN\label\exp_after:wN{sref_\l_tmpa_str}
766   \str_gset:cn {sref_\l_tmpa_str _type}\c__stex_refs_ref_str

```

```

767 }
768 }
769 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
770   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
771 }

```

20.3 Using References

```

772 \keys_define:nn { stex / sref } {
773   linktext      .tl_set:N = \l__stex_refs_linktext_tl ,
774   fallback      .tl_set:N = \l__stex_refs_fallback_tl ,
775   pre           .tl_set:N = \l__stex_refs_pre_tl ,
776   post          .tl_set:N = \l__stex_refs_post_tl ,
777   indoc         .str_set_x:N = \l__stex_refs_repo_str ,
778 }
779
780 \cs_new_protected:Nn \__stex_refs_args:n {
781   \tl_clear:N \l__stex_refs_linktext_tl
782   \tl_clear:N \l__stex_refs_fallback_tl
783   \tl_clear:N \l__stex_refs_pre_tl
784   \tl_clear:N \l__stex_refs_post_tl
785   \str_clear:N \l__stex_refs_repo_str
786   \keys_set:nn { stex / sref } { #1 }
787 }
788
789 \</package>

```

Chapter 21

STEX -Modules Implementation

```
790 <*package>
791
792 %%%%%%%%%%% modules.dtx %%%%%%%%%%%
793
794 <@@=stex_modules>
795
796   Warnings and error messages
797   \msg_new:nnn{stex}{error/unknownmodule}{
798     No~module~#1~found
799   }
800   \msg_new:nnn{stex}{error/syntax}{
801     Syntax~error:~#1
802   }
803   \msg_new:nnn{stex}{error/siglanguage}{
804     Module~#1~declares~signature~#2,~but~does~not~
805     declare~its~language
806   }
```

`\l_stex_current_module_prop` The current module:

```
805 \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

```
806 \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`

```
807 \seq_new:N \g_stex_modules_in_file_seq
808 \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
809 \prg_new_conditional:Nnn \stex_if_in_module: {p, T, F, TF} {
810   \prop_if_empty:NTF \l_stex_current_module_prop
811   \prg_return_false: \prg_return_true:
812 }

```

(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
813 \prg_new_conditional:Nnn \stex_if_module_exists:n {p, T, F, TF} {
814   \prop_if_exist:cTF { c_stex_module_#1_prop }
815   \prg_return_true: \prg_return_false:
816 }

```

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

```

\stex_add_to_current_module:n
\STEXexport
817 \cs_new_protected:Nn \stex_add_to_current_module:n {
818   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
819   \tl_put_right:Nn \l_tmpa_tl { #1 }
820   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
821 }
822 \cs_new_protected:Npn \STEXexport {
823   \begingroup
824   \newlinechar=-1\relax
825   \endlinechar=-1\relax
826   %\catcode'\ = 9\relax
827   \expandafter\endgroup\STEXexport:n
828 }
829 \cs_new_protected:Nn \STEXexport:n {
830   \ignorespaces #1
831   \stex_add_to_current_module:n { \ignorespaces #1 }
832   \stex_smsmode_set_codes:
833 }
834 \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
835 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
836   \str_set:Nx \l_tmpa_str { #1 }
837   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
838   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
839   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
840 }

```

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

```

\stex_add_import_to_current_module:n
841 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
842   \str_set:Nx \l_tmpa_str { #1 }
843   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
844   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
845   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
846 }

```


(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).

```

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

```

(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`

```

876 \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.

```

877 \cs_new_protected:Nn \stex_modules_current_namespace: {
878   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
879     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
880   }{
881     % split off file extension
882     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
883     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
884     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
885     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
886     \seq_put_right:No \l_tmpa_seq \l_tmpb_str

```

```

887     \str_set:Nx \l_stex_modules_ns_str {
888         file:/\stex_path_to_string:N \l_tmpa_seq
889     }
890 }
891 }

```

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

21.1 The module environment

module arguments:

```

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

```

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

```

915 \cs_new_protected:Nn \stex_module_setup:nn {
916     \str_set:Nx \l_stex_module_name_str { #2 }
917     \__stex_modules_args:n { #1 }
918
919     First, we set up the name and namespace of the module.
920     Are we in a nested module?
921
922     \stex_if_in_module:TF {
923         % Nested module
924         \prop_get:NnN \l_stex_current_module_prop
925         { ns } \l_stex_module_ns_str
926         \str_set:Nx \l_stex_module_name_str {
927             \prop_item:Nn \l_stex_current_module_prop
928             { name } / \l_stex_module_name_str
929         }
930     }
931 }{
932     % not nested:
933     \str_if_empty:NT \l_stex_module_ns_str {

```

```

929 \stex_modules_current_namespace:
930 \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
931 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
932 / {\l_stex_module_ns_str}
933 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
934 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
935   \str_set:Nx \l_stex_module_ns_str {
936     \stex_path_to_string:N \l_tmpa_seq
937   }
938 }
939 }
940 }

```

Next, we determine the language of the module:

```

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

```

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

```

963 \str_if_empty:NTF \l_stex_module_sig_str {
964   \str_clear:N \l_tmpa_str
965   \seq_clear:N \l_tmpa_seq
966   \tl_clear:N \l_tmpa_tl
967   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
968     name      = \l_stex_module_name_str ,
969     ns        = \l_stex_module_ns_str ,
970     imports   = \exp_not:o { \l_tmpa_seq } ,
971     constants = \exp_not:o { \l_tmpa_seq } ,
972     content   = \exp_not:o { \l_tmpa_tl } ,
973     file      = \exp_not:o { \g_stex_currentfile_seq } ,
974     lang      = \l_stex_module_lang_str ,
975     sig       = \l_stex_module_sig_str ,
976     meta      = \l_stex_module_meta_str
977   }

```

```

978 }{
979   \str_if_empty:NT \l_stex_module_lang_str {
980     \msg_error:nnnn{stex}{error/siglanguage}{
981       \l_stex_module_ns_str?\l_stex_module_name_str
982     }\l_stex_module_sig_str}
983   }
984
985   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
986   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
987   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
988   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
989   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
990   \str_set:Nx \l_tmpa_str {
991     \stex_path_to_string:N \l_tmpa_seq /
992     \l_tmpa_str . \l_stex_module_sig_str .tex
993   }
994   \IfFileExists \l_tmpa_str {
995     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
996       \seq_clear:N \l_stex_all_modules_seq
997       \prop_clear:N \l_stex_current_module_prop
998       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
999       \input { \l_tmpa_str }
1000     }
1001   }{
1002     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
1003   }
1004   \stex_activate_module:n {
1005     \l_stex_module_ns_str ? \l_stex_module_name_str
1006   }
1007   \prop_set_eq:Nc \l_stex_current_module_prop {
1008     c_stex_module_
1009     \l_stex_module_ns_str ?
1010     \l_stex_module_name_str
1011     _prop
1012   }
1013 }

```

We load the metatheory:

```

1014 \str_if_empty:NT \l_stex_module_meta_str {
1015   \str_set:Nx \l_stex_module_meta_str {
1016     \c_stex_metatheory_ns_str ? Metatheory
1017   }
1018 }
1019 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1020   \exp_args:Nx \stex_add_to_current_module:n {
1021     \stex_activate_module:n {\l_stex_module_meta_str}
1022   }
1023   \stex_activate_module:n {\l_stex_module_meta_str}
1024 }
1025 }

```

(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}

1026 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1027   \stex_reactivate_macro:N \STEXexport
1028   \stex_reactivate_macro:N \importmodule
1029   \stex_reactivate_macro:N \symdecl
1030   \stex_reactivate_macro:N \notation
1031   \stex_reactivate_macro:N \symdef
1032   \stex_module_setup:nn{#1}{#2}
1033
1034   \stex_debug:nn{modules}{
1035     New~module:\\
1036     Namespace:~\l_stex_module_ns_str\\
1037     Name:~\l_stex_module_name_str\\
1038     Language:~\l_stex_module_lang_str\\
1039     Signature:~\l_stex_module_sig_str\\
1040     Metatheory:~\l_stex_module_meta_str\\
1041     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1042   }
1043
1044   \seq_put_right:Nx \l_stex_all_modules_seq {
1045     \l_stex_module_ns_str ? \l_stex_module_name_str
1046   }
1047
1048   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1049     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1050
1051   \stex_if_smsmode:TF {
1052     \stex_smsmode_set_codes:
1053   } {
1054     \begin{stex_annotate_env} {theory} {
1055       \l_stex_module_ns_str ? \l_stex_module_name_str
1056     }
1057
1058     \stex_annotate_invisible:nnn{header}{} {
1059       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1060       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1061       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1062         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1063       }
1064     }
1065   }
1066   % TODO: Inherit metatheory for nested modules?
1067 }
1068 \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}

1069 \cs_new_protected:Nn \__stex_modules_end_module: {
1070   \str_set:Nx \l_tmpa_str {
1071     c_stex_module_
1072     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1073     \prop_item:Nn \l_stex_current_module_prop { name }
1074     _prop

```

```

1075 }
1076 %^^A \prop_new:c { \l_tmpa_str }
1077 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1078 \stex_debug:nn{modules}{Closing~module~\prop_item:Nn \l_stex_current_module_prop { name }}
1079 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1080 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1081 \NewDocumentEnvironment { @module } { 0{} m } {
1082   \par
1083   \_stex_modules_begin_module:nn{#1}{#2}
1084 } {
1085   \_stex_modules_end_module:
1086   \stex_if_smsmode:TF {
1087     \exp_args:Nx \stex_add_to_sms:n {
1088       \prop_gset_from_keyval:cn {
1089         c_stex_module_
1090         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1091         \prop_item:Nn \l_stex_current_module_prop { name }
1092         _prop
1093       } {
1094         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1095         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1096         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1097         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1098         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1099         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1100         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1101         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1102         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1103       }
1104     }
1105   }{
1106     \end{stex_annotate_env}
1107   }
1108 }

```

\stex_modules_heading: Code for document headers

```

1109 \cs_if_exist:NTF \thesection {
1110   \newcounter{module}[section]
1111 }{
1112   \newcounter{module}
1113 }
1114
1115 \bool_if:NT \c_stex_showmods_bool {
1116   \latexml_if:F { \RequirePackage{mdframed} }
1117 }
1118
1119 \cs_new_protected:Nn \stex_modules_heading: {
1120   \stepcounter{module}
1121   \par
1122   \bool_if:NT \c_stex_showmods_bool {

```

```

1123 \noindent{\textbf{Module} ~
1124 \cs_if_exist:NT \thesection {\thesection.}
1125 \themodule ~ [\l_stex_module_name_str]
1126 }
1127 \str_if_empty:NTF \l_stex_module_title_str {
1128 }{
1129 \quad(\l_stex_module_title_str)\hfill
1130 }\par
1131 }
1132 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1133 % TODO
1134 \stex_ref_new_doc_target:n \l_stex_module_name_str
1135 }

```

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

Finally:

```

1136 \NewDocumentEnvironment { module } { 0 } { m } {
1137 \bool_if:NT \c_stex_showmods_bool {
1138 \begin{mdframed}
1139 }
1140 \begin{@module}[#1]{#2}
1141 \stex_modules_heading:
1142 }{
1143 \end{@module}
1144 \bool_if:NT \c_stex_showmods_bool {
1145 \end{mdframed}
1146 }
1147 }

```

21.2 Invoking modules

`\STEXModule`
`\stex_invoke_module:n`

```

1148 \NewDocumentCommand \STEXModule { m } {
1149 \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1150 \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1151 \tl_set:Nn \l_tmpa_tl {
1152 \msg_error:nnn{stex}{error/unknownmodule}{#1}
1153 }
1154 \seq_map_inline:Nn \l_stex_all_modules_seq {
1155 \str_set:Nn \l_tmpb_str { ##1 }
1156 \str_if_eq:eeT { \l_tmpa_str } {
1157 \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1158 } {
1159 \seq_map_break:n {
1160 \tl_set:Nn \l_tmpa_tl {
1161 \stex_invoke_module:n { ##1 }
1162 }
1163 }
1164 }
1165 }
1166 \l_tmpa_tl
1167 }
1168

```

```

1169 \cs_new_protected:Nn \stex_invoke_module:n {
1170   \stex_debug:nn{modules}{Invoking~module~#1}
1171   \peek_charcode_remove:NTF ! {
1172     \__stex_modules_invoke_uri:nN { #1 }
1173   } {
1174     \peek_charcode_remove:NTF ? {
1175       \__stex_modules_invoke_symbol:nn { #1 }
1176     } {
1177       \msg_error:nnn{stex}{error/syntax}{
1178         ?~or~!~expected~after~
1179         \c_backslash_str STEXModule{#1}
1180       }
1181     }
1182   }
1183 }
1184
1185 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1186   \str_set:Nn #2 { #1 }
1187 }
1188
1189 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1190   \stex_invoke_symbol:n{#1?#2}
1191 }

```

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

`\stex_activate_module:n`

```

1192 \cs_new_protected:Nn \stex_activate_module:n {
1193   \stex_debug:nn{modules}{Activating~module~#1}
1194   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1195     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1196     \prop_item:cn { c_stex_module_#1_prop } { content }
1197   }
1198 }

```

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

```

1199 </package>

```


Chapter 22

STEX -Module Inheritance Implementation

```
1200 <*package>
1201
1202 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1203
```

22.1 SMS Mode

```
1204 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq
1205 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1206 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1207 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1208
1209 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1210   \makeatletter
1211   \makeatother
1212   \ExplSyntaxOn
1213   \ExplSyntaxOff
1214 }
1215
1216 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1217   \symdef
1218   \importmodule
1219   \notation
1220   \symdecl
1221   \STEXexport
1222 }
1223
1224 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1225   \tl_to_str:n {
1226     module,
1227     @module
```

```

1228 }
1229 }

```

(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
1230 \bool_new:N \g__stex_smsmode_bool
1231 \bool_set_false:N \g__stex_smsmode_bool
1232 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1233   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1234 }

```

(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
1235 \bool_new:N \g__stex_smsmode_catcode_bool
1236 \bool_set_false:N \g__stex_smsmode_catcode_bool
1237 \prg_new_conditional:Nnn \_stex_smsmode_if_catcodes: { p, T, F, TF } {
1238   \bool_if:NTF \g__stex_smsmode_catcode_bool
1239   \prg_return_true: \prg_return_false:
1240 }

```

(End definition for `_stex_smsmode_if_catcodes:TF`.)

`\stex_smsmode_set_codes:`

```

1241 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1242   \stex_if_smsmode:T {
1243     \_stex_smsmode_if_catcodes:F {
1244       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1245       \exp_after:wN \char_gset_active_eq:NN
1246       \c_backslash_str \_stex_smsmode_cs:
1247       \tex_global:D \char_set_catcode_active:N \
1248       \tex_global:D \char_set_catcode_other:N $
1249       \tex_global:D \char_set_catcode_other:N ^
1250       \tex_global:D \char_set_catcode_other:N _
1251       \tex_global:D \char_set_catcode_other:N &
1252       \tex_global:D \char_set_catcode_other:N ##
1253     }
1254   }
1255 } \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.

```

1256 \cs_new_protected:Nn \_stex_smsmode_unset_codes: {
1257   \_stex_smsmode_if_catcodes:T {
1258     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1259     \exp_after:wN \tex_global:D \exp_after:wN
1260     \char_set_catcode_escape:N \c_backslash_str
1261     \tex_global:D \char_set_catcode_math_toggle:N $
1262     \tex_global:D \char_set_catcode_math_superscript:N ^
1263     \tex_global:D \char_set_catcode_math_subscript:N _
1264     \tex_global:D \char_set_catcode_alignment:N &
1265     \tex_global:D \char_set_catcode_parameter:N ##
1266   }
1267 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1268 \cs_new_protected:Nn \stex_in_smsmode:nn {
1269   \vbox_set:Nn \l_tmpa_box {
1270     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1271     \bool_gset_true:N \g__stex_smsmode_bool
1272     \stex_smsmode_set_codes:
1273     #2
1274     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1275     \stex_if_smsmode:F {
1276       \__stex_smsmode_unset_codes:
1277     }
1278   }
1279   \box_clear:N \l_tmpa_box
1280 }

```

(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:

```

1281 \cs_new_protected:Nn \_stex_smsmode_cs: {
1282   \str_clear:N \l_tmpa_str
1283   \peek_analysis_map_inline:n {
1284     % #1: token (one expansion)
1285     % #2: charcode
1286     % #3 catcode
1287     \token_if_eq_charcode:NNTF ##3 B {
1288       % token is a letter
1289       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1290     } {
1291       \str_if_empty:NTF \l_tmpa_str {
1292         % we don't allow (or need) single non-letter CSs
1293         % for now
1294         \peek_analysis_map_break:
1295       }{
1296         \str_if_eq:onTF \l_tmpa_str { begin } {
1297           \peek_analysis_map_break:n {
1298             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1299           }
1300         } {
1301           \str_if_eq:onTF \l_tmpa_str { end } {
1302             \peek_analysis_map_break:n {
1303               \exp_after:wN \_stex_smsmode_checkend:n ##1
1304             }
1305           } {
1306             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1307             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1308               \g_stex_smsmode_allowedmacros_tl
1309               { \use:c{\l_tmpa_str} } {
1310               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1311               \peek_analysis_map_break:n {
1312                 \exp_after:wN \l_tmpa_tl ##1
1313               }

```

```

1314     } {
1315         \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1316         \g_stex_smsmode_allowedmacros_escape_tl
1317         { \use:c{\l_tmpa_str} } {
1318             \__stex_smsmode_unset_codes:
1319             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1320             % TODO \__stex_smsmode_rescan_cs:
1321             \int_compare:nNnTF {##2} = {92} {
1322                 \peek_analysis_map_break:n {
1323                     \__stex_smsmode_unset_codes:
1324                     \__stex_smsmode_rescan_cs:
1325                 }
1326             } {
1327                 \peek_analysis_map_break:n {
1328                     \exp_after:wN \l_tmpa_tl ##1
1329                 }
1330             }
1331         } {
1332             \int_compare:nNnTF {##2} = {92} {
1333                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1334             } {
1335                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1336             }
1337         }
1338     }
1339 }
1340 }
1341 }
1342 }
1343 }
1344 }

```

(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:

```

1345 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1346     \str_clear:N \l_tmpb_str
1347     \peek_analysis_map_inline:n {
1348         \token_if_eq_charcode:NNTF ##3 B {
1349             % token is a letter
1350             \exp_args:NNNo \str_put_right:Nn \l_tmpb_str { ##1 }
1351         } {
1352             \peek_analysis_map_break:n {
1353                 \exp_after:wN \use:c \exp_after:wN {
1354                     \exp_after:wN \l_tmpa_str\exp_after:wN
1355                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1356             }
1357         }
1358     }
1359 }

```

(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.

```

1360 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1361   \str_set:Nn \l_tmpa_str { #1 }
1362   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1363     \__stex_smsmode_unset_codes:
1364     \begin{#1}
1365   }
1366 }
```

(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.

```

1367 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1368   \str_set:Nn \l_tmpa_str { #1 }
1369   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1370     \end{#1}
1371   }
1372 }
```

(End definition for `__stex_smsmode_checkend:n`.)

22.2 Inheritance

1373 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1374 \cs_new_protected:Nn \stex_import_module_uri:nn {
1375   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1376   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1377   \str_if_empty:NT \l__stex_importmodule_archive_str {
1378     \prop_if_empty:NF \l_stex_current_repository_prop {
1379       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1380     }
1381   }
1382
1383   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1384   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1385   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1386
1387   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1388     \stex_modules_current_namespace:
1389     \str_if_empty:NF \l__stex_importmodule_path_str {
1390       \str_set:Nx \l_stex_module_ns_str {
1391         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1392       }
1393     }
1394   }{
1395     \stex_require_repository:n \l__stex_importmodule_archive_str
1396     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1397     \l_stex_module_ns_str
1398     \str_if_empty:NF \l__stex_importmodule_path_str {
1399       \str_set:Nx \l_stex_module_ns_str {
1400         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1401       }
1402     }
```

```

1402     }
1403   }
1404 }

```

(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 1405 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1406 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1407 \str_new:N \l__stex_importmodule_path_str
1408 \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>}
1409 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1410   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1411
1412     % archive
1413     \str_set:Nx \l_tmpa_str { #2 }
1414     \str_if_empty:NTF \l_tmpa_str {
1415       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1416     } {
1417       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1418       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1419       \seq_put_right:Nn \l_tmpa_seq { source }
1420     }
1421
1422     % path
1423     \str_set:Nx \l_tmpb_str { #3 }
1424     \str_if_empty:NTF \l_tmpb_str {
1425       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1426
1427       \ltx@ifpackageloaded{babel} {
1428         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1429           { \language } \l_tmpb_str {
1430           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1431         }
1432       } {
1433         \str_clear:N \l_tmpb_str
1434       }
1435
1436       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1437       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1438         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1439       }{
1440         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1441         \IfFileExists{ \l_tmpa_str.tex }{
1442           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1443         }{
1444           % try english as default
1445           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1446           \IfFileExists{ \l_tmpa_str.en.tex }{
1447             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1448         }{
1449             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1450         }
1451     }
1452 }
1453
1454 } {
1455     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1456     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1457
1458     \ltx@ifpackageloaded{babel} {
1459         \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1460             { \language } \l_tmpb_str {
1461             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1462         }
1463     } {
1464         \str_clear:N \l_tmpb_str
1465     }
1466
1467     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1468
1469     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1470     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1471         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1472     }{
1473         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1474         \IfFileExists{ \l_tmpa_str/#4.tex }{
1475             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1476         }{
1477             % try english as default
1478             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1479             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1480                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1481             }{
1482                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1483                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1484                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1485                 }{
1486                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1487                     \IfFileExists{ \l_tmpa_str.tex }{
1488                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1489                     }{
1490                         % try english as default
1491                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1492                         \IfFileExists{ \l_tmpa_str.en.tex }{
1493                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1494                         }{
1495                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1496                         }
1497                     }
1498                 }
1499             }
1500         }
1501     }

```

```

1502     }
1503
1504     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1505     \seq_clear:N \g_stex_modules_in_file_seq
1506     % \exp_args:Nnx \use:nn {
1507         \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1508             \seq_clear:N \l_stex_all_modules_seq
1509             \prop_clear:N \l_stex_current_module_prop
1510             \str_set:Nx \l_tmpb_str { #2 }
1511             \str_if_empty:NF \l_tmpb_str {
1512                 \stex_set_current_repository:n { #2 }
1513             }
1514             \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1515             \input { \g__stex_importmodule_file_str }
1516         }
1517     % }{
1518
1519     % }
1520     \prop_gput:Noo \g_stex_module_files_prop
1521     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1522     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1523
1524     \stex_if_module_exists:nF { #1 ? #4 } {
1525         \msg_error:nnn{stex}{error/unknownmodule}{
1526             #1?#4~(in~file~\g__stex_importmodule_file_str)
1527         }
1528     }
1529 }
1530 \stex_activate_module:n { #1 ? #4 }
1531 }

```

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

`\importmodule`

```

1532 \NewDocumentCommand \importmodule { 0{} m } {
1533     \stex_import_module_uri:nn { #1 } { #2 }
1534     \stex_debug:nn{modules}{Importing~module:~
1535         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1536     }
1537     \stex_if_smsmode:F {
1538         \stex_import_require_module:nnnn
1539         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1540         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1541         \stex_annotate_invisible:nnn
1542         {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1543     }
1544     \exp_args:Nx \stex_add_to_current_module:n {
1545         \stex_import_require_module:nnnn
1546         { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1547         { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1548     }
1549     \exp_args:Nx \stex_add_import_to_current_module:n {
1550         \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1551     }

```



```

1552 \stex_smsmode_set_codes:
1553 }
1554 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1555 \NewDocumentCommand \usemodule { 0{} m } {
1556 \stex_if_smsmode:F {
1557 \stex_import_module_uri:nn { #1 } { #2 }
1558 \stex_import_require_module:nnnn
1559 { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1560 { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1561 \stex_annotate_invisible:nnn
1562 {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1563 }
1564 \stex_smsmode_set_codes:
1565 }

```

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

```

1566 \endpackage

```

Chapter 23

STEX -Symbols Implementation

```
1567 <*package>
1568
1569 %%%%%%%%%% symbols.dtx %%%%%%%%%%
1570
```

Warnings and error messages

```
1571
```

23.1 Symbol Declarations

```
1572 <@@=stex_symdecl>
```

`\l_stex_all_symbols_seq` Stores all available symbols

```
1573 \seq_new:N \l_stex_all_symbols_seq
```

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

`\STEXsymbol`

```
1574 \NewDocumentCommand \STEXsymbol { m } {
1575   \stex_get_symbol:n { #1 }
1576   \exp_args:No
1577   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1578 }
```

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

symdecl arguments:

```
1579 \keys_define:nn { stex / symdecl } {
1580   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1581   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1582   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1583   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1584   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1585   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1586   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1587   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1588 }
```

```

1589
1590 \bool_new:N \l_stex_symdecl_make_macro_bool
1591
1592 \cs_new_protected:Nn \__stex_symdecl_args:n {
1593   \str_clear:N \l_stex_symdecl_name_str
1594   \str_clear:N \l_stex_symdecl_args_str
1595   \bool_set_false:N \l_stex_symdecl_local_bool
1596   \tl_clear:N \l_stex_symdecl_type_tl
1597   \tl_clear:N \l_stex_symdecl_definiens_tl
1598
1599   \keys_set:nn { stex / symdecl } { #1 }
1600 }

```

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

```

1601
1602 \NewDocumentCommand \symdecl { s O{} m } {
1603   \__stex_symdecl_args:n { #2 }
1604   \IfBooleanTF #1 {
1605     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1606   } {
1607     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1608   }
1609   \stex_symdecl_do:n { #3 }
1610   \stex_smsmode_set_codes:
1611 }
1612 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1613 \cs_new_protected:Nn \stex_symdecl_do:n {
1614   \stex_if_in_module:F {
1615     % TODO throw error? some default namespace?
1616   }
1617
1618   \str_if_empty:NT \l_stex_symdecl_name_str {
1619     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1620   }
1621
1622   \prop_if_exist:cT { g_stex_symdecl_
1623     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1624     \prop_item:Nn \l_stex_current_module_prop {name} ?
1625     \l_stex_symdecl_name_str
1626     _prop
1627   }{
1628     % TODO throw error (beware of circular dependencies)
1629   }
1630
1631   \prop_clear:N \l_tmpa_prop
1632   \prop_put:Nnx \l_tmpa_prop { module } {
1633     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1634     \prop_item:Nn \l_stex_current_module_prop {name}
1635   }

```

```

1636 \seq_clear:N \l_tmpa_seq
1637 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1638 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1639 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1640 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1641
1642 \exp_args:No \stex_add_constant_to_current_module:n {
1643   \l_stex_symdecl_name_str
1644 }
1645
1646 % arity/args
1647 \int_zero:N \l_tmpb_int
1648
1649 \bool_set_true:N \l_tmpa_bool
1650 \str_map_inline:Nn \l_stex_symdecl_args_str {
1651   \token_case_meaning:NnF ##1 {
1652     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1653     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1654     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1655     {\tl_to_str:n a} {
1656       \bool_set_false:N \l_tmpa_bool
1657       \int_incr:N \l_tmpb_int
1658     }
1659     {\tl_to_str:n B} {
1660       \bool_set_false:N \l_tmpa_bool
1661       \int_incr:N \l_tmpb_int
1662     }
1663   }{
1664     \msg_set:nnn{stex}{error/wrongargs}{
1665       args~value~in~symbol~declaration~for~
1666       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1667       \prop_item:Nn \l_stex_current_module_prop {name} ?
1668       \l_stex_symdecl_name_str ~
1669       needs~to~be~
1670       i,~a,~b~or~B,~but~##1~given
1671     }
1672     \msg_error:nn{stex}{error/wrongargs}
1673   }
1674 }
1675 \bool_if:NTF \l_tmpa_bool {
1676   % possibly numeric
1677   \str_if_empty:NTF \l_stex_symdecl_args_str {
1678     \prop_put:Nnn \l_tmpa_prop { args } {}
1679     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1680   }{
1681     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1682     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1683     \str_clear:N \l_tmpa_str
1684     \int_step_inline:nn \l_tmpa_int {
1685       \str_put_right:Nn \l_tmpa_str i
1686     }
1687     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1688   }
1689 } {

```

```

1690 \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1691 \prop_put:Nnx \l_tmpa_prop { arity }
1692 { \str_count:N \l_stex_symdecl_args_str }
1693 }
1694 \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1695
1696
1697 % semantic macro
1698
1699 \bool_if:NT \l_stex_symdecl_make_macro_bool {
1700 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1701 \prop_item:Nn \l_tmpa_prop { module } ?
1702 \prop_item:Nn \l_tmpa_prop { name }
1703 } }
1704
1705 \bool_if:NF \l_stex_symdecl_local_bool {
1706 \exp_args:Nx \stex_add_to_current_module:n {
1707 \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1708 \prop_item:Nn \l_tmpa_prop { module } ?
1709 \prop_item:Nn \l_tmpa_prop { name }
1710 } }
1711 }
1712 }
1713 }
1714
1715 % add to all symbols
1716
1717 \bool_if:NF \l_stex_symdecl_local_bool {
1718 \exp_args:Nx \stex_add_to_current_module:n {
1719 \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1720 \prop_item:Nn \l_tmpa_prop { module } ?
1721 \prop_item:Nn \l_tmpa_prop { name }
1722 }
1723 }
1724 }
1725
1726 \stex_debug:nn{symbols}{New~symbol:~
1727 \prop_item:Nn \l_tmpa_prop { module } ?
1728 \prop_item:Nn \l_tmpa_prop { name } ^^J
1729 Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1730 Args:~\prop_item:Nn \l_tmpa_prop { args }
1731 }
1732
1733 % circular dependencies require this:
1734
1735 \prop_if_exist:cF {
1736 g_stex_symdecl_
1737 \prop_item:Nn \l_tmpa_prop { module } ?
1738 \prop_item:Nn \l_tmpa_prop { name }
1739 _prop
1740 } {
1741 \prop_gset_eq:cN {
1742 g_stex_symdecl_
1743 \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1744     \prop_item:Nn \l_tmpa_prop { name }
1745     _prop
1746   } \l_tmpa_prop
1747 }
1748
1749 \stex_if_smsmode:TF {
1750   \bool_if:NF \l_stex_symdecl_local_bool {
1751     \exp_args:Nx \stex_add_to_sms:n {
1752       \prop_gset_from_keyval:cn {
1753         g_stex_symdecl_
1754         \prop_item:Nn \l_tmpa_prop { module } ?
1755         \prop_item:Nn \l_tmpa_prop { name }
1756         _prop
1757       } {
1758         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1759         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1760         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1761         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1762         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1763         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1764         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1765         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1766       }
1767       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1768         \prop_item:Nn \l_tmpa_prop { module } ?
1769         \prop_item:Nn \l_tmpa_prop { name }
1770       }
1771     }
1772   }
1773 }{
1774   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1775     \prop_item:Nn \l_tmpa_prop { module } ?
1776     \prop_item:Nn \l_tmpa_prop { name }
1777   }
1778   \stex_if_do_html:T {
1779     \stex_annotate_invisible:nnn {symdecl} {
1780       \prop_item:Nn \l_tmpa_prop { module } ?
1781       \prop_item:Nn \l_tmpa_prop { name }
1782     } {
1783       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1784       \stex_annotate_invisible:nnn{args}{}{
1785         \prop_item:Nn \l_tmpa_prop { args }
1786       }
1787       \stex_annotate_invisible:nnn{macroname}{}{#1}
1788       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1789         \stex_annotate_invisible:nnn{definiens}{}
1790         {\l_stex_symdecl_definiens_tl$}
1791       }
1792     }
1793   }
1794 }
1795 }

```

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

`\stex_get_symbol:n`

```
1796 \str_new:N \l_stex_get_symbol_uri_str
1797
1798 \cs_new_protected:Nn \stex_get_symbol:n {
1799   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1800     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1801   }{
1802     % argument is a string
1803     % is it a command name?
1804     \cs_if_exist:cTF { #1 }{
1805       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1806       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1807       \str_if_empty:NNTF \l_tmpa_str {
1808         \exp_args:Nx \cs_if_eq:NNTF {
1809           \tl_head:N \l_tmpa_tl
1810         } \stex_invoke_symbol:n {
1811           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1812         }{
1813           \__stex_symdecl_get_symbol_from_string:n { #1 }
1814         }
1815       } {
1816         \__stex_symdecl_get_symbol_from_string:n { #1 }
1817       }
1818     }{
1819       % argument is not a command name
1820       \__stex_symdecl_get_symbol_from_string:n { #1 }
1821       % \l_stex_all_symbols_seq
1822     }
1823   }
1824 }
1825
1826 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1827   \str_set:Nn \l_tmpa_str { #1 }
1828   \bool_set_false:N \l_tmpa_bool
1829   \stex_if_in_module:T {
1830     \prop_get:NnN \l_stex_current_module_prop
1831     { constants } \l_tmpa_seq
1832     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1833       \bool_set_true:N \l_tmpa_bool
1834       \str_set:Nx \l_stex_get_symbol_uri_str {
1835         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1836         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1837       }
1838     }
1839   }
1840   \bool_if:NF \l_tmpa_bool {
1841     \tl_set:Nn \l_tmpa_tl {
1842       \msg_set:nnn{stex}{error/unknownsymbol}{
1843         No~symbol~#1~found!
1844       }
1845     }
1846     \msg_error:nn{stex}{error/unknownsymbol}
1847   }
1848   \str_set:Nn \l_tmpa_str { #1 }
1849   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1849 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1850   \str_set:Nn \l_tmpb_str { ##1 }
1851   \str_if_eq:eeT { \l_tmpa_str } {
1852     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1853   } {
1854     \seq_map_break:n {
1855       \tl_set:Nn \l_tmpa_tl {
1856         \str_set:Nn \l_stex_get_symbol_uri_str {
1857           ##1
1858         }
1859       }
1860     }
1861   }
1862 }
1863 \l_tmpa_tl
1864 }
1865 }
1866
1867 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1868   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1869   { \tl_tail:N \l_tmpa_tl }
1870   \tl_if_single:NTF \l_tmpa_tl {
1871     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1872       \exp_after:wN \str_set:Nn \exp_after:wN
1873       \l_stex_get_symbol_uri_str \l_tmpa_tl
1874     }{
1875       % TODO
1876       % tail is not a single group
1877     }
1878   }{
1879     % TODO
1880     % tail is not a single group
1881   }
1882 }

```

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

23.2 Notations

```

1883 <@@=stex_notation>
1884 notation arguments:
1885 \keys_define:nn { stex / notation } {
1886   lang .tl_set_x:N = \l__stex_notation_lang_str ,
1887   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1888   prec .tl_set_x:N = \l__stex_notation_prec_str ,
1889   op .tl_set:N = \l__stex_notation_op_tl ,
1890   unknown .code:n = \str_set:Nx
1891     \l__stex_notation_variant_str \l_keys_key_str
1892 }
1893
1894 \cs_new_protected:Nn \__stex_notation_args:n {
1895   \str_clear:N \l__stex_notation_lang_str
1896   \str_clear:N \l__stex_notation_variant_str

```



```

1896 \str_clear:N \l__stex_notation_prec_str
1897 \tl_clear:N \l__stex_notation_op_tl
1898
1899 \keys_set:nn { stex / notation } { #1 }
1900 }

```

\notation

```

1901 \NewDocumentCommand \notation { 0{ } m } {
1902   \__stex_notation_args:n { #1 }
1903   \tl_clear:N \l_stex_symdecl_definiens_tl
1904   \stex_get_symbol:n { #2 }
1905   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1906 }
1907 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

1908 \cs_new_protected:Nn \stex_notation_do:nn {
1909   \prop_set_eq:Nc \l_tmpa_prop {
1910     g_stex_symdecl_ #1 _prop
1911   }
1912
1913   \prop_clear:N \l_tmpb_prop
1914   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1915   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1916   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1917
1918   % precedences
1919   \seq_clear:N \l_tmpb_seq
1920   \exp_args:NNno
1921   \str_if_empty:NTF \l__stex_notation_prec_str {
1922     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1923     \int_compare:nNnTF \l_tmpa_str = 0 {
1924       \exp_args:NNnx
1925       \prop_put:Nno \l_tmpb_prop { opprec }
1926       { \neginfprec }
1927     }{
1928       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1929     }
1930   } {
1931     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1932       \exp_args:NNnx
1933       \prop_put:Nno \l_tmpb_prop { opprec }
1934       { \neginfprec }
1935       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1936       \int_step_inline:nn { \l_tmpa_str } {
1937         \exp_args:NNx
1938         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1939       }
1940     }{
1941       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1942       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1943         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1944         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1945         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1946         \l_tmpa_seq {\tl_to_str:n{x}} { \l_tmpa_str }
1947         \seq_map_inline:Nn \l_tmpa_seq {
1948             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1949         }
1950     }
1951     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1952 }{
1953     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1954     \int_compare:nNnTF \l_tmpa_str = 0 {
1955         \exp_args:NNnx
1956         \prop_put:Nno \l_tmpb_prop { opprec }
1957         { \infprec }
1958     }{
1959         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1960     }
1961 }
1962 }
1963 }
1964
1965 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1966 \int_step_inline:nn { \l_tmpa_str } {
1967     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1968         \exp_args:NNx
1969         \seq_put_right:Nn \l_tmpb_seq {
1970             \prop_item:Nn \l_tmpb_prop { opprec }
1971         }
1972     }
1973 }
1974
1975 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1976 \tl_clear:N \l_tmpa_tl
1977
1978 \int_compare:nNnTF \l_tmpa_str = 0 {
1979     \exp_args:NNe
1980     \cs_set:Npn \l__stex_notation_macrocode_cs {
1981         \_stex_term_math_oms:nnnn { #1 }
1982         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1983         { \prop_item:Nn \l_tmpb_prop { opprec } }
1984         { \exp_not:n { #2 } }
1985     }
1986     \__stex_notation_final:
1987 }{
1988     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1989     \str_if_in:NnTF \l_tmpb_str b {
1990         \exp_args:Nne \use:nn
1991         {
1992             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1993             \cs_set:Npn \l_tmpa_str { {
1994                 \_stex_term_math_omb:nnnn { #1 }
1995                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1996                 { \prop_item:Nn \l_tmpb_prop { opprec } }
1997                 { \exp_not:n { #2 } }
1998             }}

```

```

1999   }{
2000     \str_if_in:NnTF \l_tmpb_str B {
2001       \exp_args:Nne \use:nn
2002       {
2003         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2004         \cs_set:Npn \l_tmpa_str } { {
2005           \_stex_term_math_omb:nnnn { #1 }
2006           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2007           { \prop_item:Nn \l_tmpb_prop { opprec } }
2008           { \exp_not:n { #2 } }
2009         } }
2010       }{
2011         \exp_args:Nne \use:nn
2012         {
2013           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2014           \cs_set:Npn \l_tmpa_str } { {
2015             \_stex_term_math_oma:nnnn { #1 }
2016             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2017             { \prop_item:Nn \l_tmpb_prop { opprec } }
2018             { \exp_not:n { #2 } }
2019           } }
2020         }
2021       }
2022
2023       \int_zero:N \l_tmpa_int
2024       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2025       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2026       \__stex_notation_arguments:
2027     }
2028   }

```

(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

```

2029 \cs_new_protected:Nn \__stex_notation_arguments: {
2030   \int_incr:N \l_tmpa_int
2031   \str_if_empty:NnTF \l_tmpa_str {
2032     \__stex_notation_final:
2033   }{
2034     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2035     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2036     \str_if_eq:VnTF \l_tmpb_str a {
2037       \__stex_notation_argument_assoc:n
2038     }{
2039       \str_if_eq:VnTF \l_tmpb_str B {
2040         \__stex_notation_argument_assoc:n
2041       }{
2042         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2043         \tl_put_right:Nx \l_tmpa_tl {
2044           { \_stex_term_math_arg:nnn
2045             { \int_use:N \l_tmpa_int }
2046             { \l_tmpb_str }
2047             { ####\int_use:N \l_tmpa_int }
2048           }

```

```

2049     }
2050     \__stex_notation_arguments:
2051   }
2052 }
2053 }
2054 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2055 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2056   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2057   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2058   \tl_put_right:Nx \l_tmpa_tl {
2059     { \stex_term_math_assoc_arg:nnnn
2060       { \int_use:N \l_tmpa_int }
2061       { \l_tmpb_str }
2062       \exp_args:No \exp_not:n
2063       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2064       { ####\int_use:N \l_tmpa_int }
2065     }
2066   }
2067   \__stex_notation_arguments:
2068 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2069 \cs_new_protected:Nn \__stex_notation_final: {
2070   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2071   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2072   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2073   \exp_args:Nne \use:nn
2074   {
2075     \cs_generate_from_arg_count:cNnn {
2076       stex_notation_ \l_tmpa_str \c_hash_str
2077       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2078       _cs
2079     }
2080     \cs_gset:Npn \l_tmpb_str } { {
2081       \exp_after:wN \exp_after:wN \exp_after:wN
2082       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2083       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2084     } }
2085
2086   \tl_if_empty:NF \l__stex_notation_op_tl {
2087     \cs_gset:cpx {
2088       stex_op_notation_ \l_tmpa_str \c_hash_str
2089       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2090       _cs
2091     } {
2092       \stex_term_oms:nnn {
2093         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2094         \l__stex_notation_lang_str

```

```

2095     }{
2096         \l_tmpa_str
2097     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2098 }
2099 }
2100
2101
2102
2103 \stex_debug:nn{symbols}{
2104     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2105     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2106     Operator~precedence:~
2107     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2108     Argument~precedences:~
2109     \seq_use:Nn \l_tmpa_seq {,~}^^J
2110     Notation: \cs_meaning:c {
2111         stex_notation_ \l_tmpa_str \c_hash_str
2112         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2113         _cs
2114     }
2115 }
2116
2117 \prop_gset_eq:cN {
2118     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2119     \c_hash_str \l__stex_notation_lang_str _prop
2120 } \l_tmpb_prop
2121
2122 \exp_args:Nx
2123 \stex_add_to_current_module:n {
2124     \prop_get:cnN {
2125         g_stex_symdecl_
2126         \prop_item:Nn \l_tmpb_prop { symbol }
2127         _prop
2128     } { notations } \exp_not:N \l_tmpa_seq
2129     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2130         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2131     }
2132     \prop_put:cno {
2133         g_stex_symdecl_
2134         \prop_item:Nn \l_tmpb_prop { symbol }
2135         _prop
2136     } { notations } \exp_not:N \l_tmpa_seq
2137 }
2138
2139 \stex_if_smsmode:TF {
2140     \stex_smsmode_set_codes:
2141     \exp_args:Nx \stex_add_to_sms:n {
2142         \prop_gset_from_keyval:cn {
2143             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2144             \c_hash_str \l__stex_notation_lang_str _prop
2145         } {
2146             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2147             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2148             variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2149         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2150         argprec      = \prop_item:Nn \l_tmpb_prop { argprec }      ,
2151     }
2152 }
2153 }{
2154   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2155   \seq_put_right:Nx \l_tmpa_seq {
2156     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2157   }
2158   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2159   \prop_set_eq:cN {
2160     g_stex_symdecl_ \l_tmpa_str _prop
2161   } \l_tmpa_prop
2162
2163   % HTML annotations
2164   \stex_if_do_html:T {
2165     \stex_annotate_invisible:nnn { notation }
2166     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2167       \stex_annotate_invisible:nnn { notationfragment }
2168       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2169     }
2170     \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2171     \stex_annotate_invisible:nnn { precedence }
2172     { \prop_item:Nn \l_tmpb_prop { opprec } ;
2173       \seq_use:Nn \l_tmpa_seq { x }
2174     }{}
2175
2176     \int_zero:N \l_tmpa_int
2177     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2178     \tl_clear:N \l_tmpa_tl
2179     \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{
2180       \int_incr:N \l_tmpa_int
2181       \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2182       \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2183       \str_if_eq:VnTF \l_tmpb_str a {
2184         \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2185           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2186           \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2187         } }
2188       }{
2189         \str_if_eq:VnTF \l_tmpb_str B {
2190           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2191             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2192             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2193           } }
2194         }{
2195           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2196             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2197           } }
2198         }
2199       }
2200     }
2201     \stex_annotate_invisible:nnn { notationcomp }{}{
2202       $ \exp_args:Nno \use:nn { \use:c {
2203         stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2203         \c_hash_str \l__stex_notation_variant_str
2204         \c_hash_str \l__stex_notation_lang_str _cs
2205     } } { \l_tmpa_tl } $
2206   }
2207 }
2208 }
2209 }
2210 }

```

(End definition for _stex_notation_final:.)

\symdef

```

2211 \keys_define:nn { stex / symdef } {
2212   name .str_set_x:N = \l_stex_symdecl_name_str ,
2213   local .bool_set:N = \l_stex_symdecl_local_bool ,
2214   args .str_set_x:N = \l_stex_symdecl_args_str ,
2215   type .tl_set:N = \l_stex_symdecl_type_tl ,
2216   def .tl_set:N = \l_stex_symdecl_definiens_tl ,
2217   op .tl_set:N = \l__stex_notation_op_tl ,
2218   lang .str_set_x:N = \l__stex_notation_lang_str ,
2219   variant .str_set_x:N = \l__stex_notation_variant_str ,
2220   prec .str_set_x:N = \l__stex_notation_prec_str ,
2221   unknown .code:n = \str_set:Nx
2222     \l__stex_notation_variant_str \l_keys_key_str
2223 }
2224
2225 \cs_new_protected:Nn \_stex_notation_symdef_args:n {
2226   \str_clear:N \l_stex_symdecl_name_str
2227   \str_clear:N \l_stex_symdecl_args_str
2228   \bool_set_false:N \l_stex_symdecl_local_bool
2229   \tl_clear:N \l_stex_symdecl_type_tl
2230   \tl_clear:N \l_stex_symdecl_definiens_tl
2231   \str_clear:N \l__stex_notation_lang_str
2232   \str_clear:N \l__stex_notation_variant_str
2233   \str_clear:N \l__stex_notation_prec_str
2234   \tl_clear:N \l__stex_notation_op_tl
2235
2236   \keys_set:nn { stex / symdef } { #1 }
2237 }
2238
2239 \NewDocumentCommand \symdef { 0{} m } {
2240   \_stex_notation_symdef_args:n { #1 }
2241   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2242   \stex_symdecl_do:n { #2 }
2243   \exp_args:Nx \stex_notation_do:nn {
2244     \prop_item:Nn \l_tmpa_prop { module } ?
2245     \prop_item:Nn \l_tmpa_prop { name }
2246   }
2247 }
2248 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2249 \endpackage

```

Chapter 24

STEX -Terms Implementation

```
2250 <*package>
2251
2252 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2253
2254 <@@=stex_terms>
2255
2256 Warnings and error messages
2257 \msg_new:nnn{stex}{error/nonotation}{
2258   Symbol~#1~invoked,~but~has~no~notation#2!
2259 }
2260 \msg_new:nnn{stex}{error/notationarg}{
2261   Error~in~parsing~notation~#1
2262 }
```

24.1 Symbol Invocations

Arguments:

```
2262 \keys_define:nn { stex / terms } {
2263   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2264   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2265   unknown .code:n = \str_set:Nx
2266     \l__stex_terms_variant_str \l_keys_key_str
2267 }
2268
2269 \cs_new_protected:Nn \__stex_terms_args:n {
2270   \str_clear:N \l__stex_terms_lang_str
2271   \str_clear:N \l__stex_terms_variant_str
2272   \str_clear:N \l__stex_terms_prec_str
2273   \tl_clear:N \l__stex_terms_op_tl
2274
2275   \keys_set:nn { stex / terms } { #1 }
2276 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro


```

2277 \cs_new_protected:Nn \stex_invoke_symbol:n {
2278   \if_mode_math:
2279     \exp_after:wN \__stex_terms_invoke_math:n
2280   \else:
2281     \exp_after:wN \__stex_terms_invoke_text:n
2282   \fi: { #1 }
2283 }

```

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

`__stex_terms_invoke_math:n`

```

2284 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2285   \peek_charcode_remove:NTF ! {
2286     \peek_charcode:NTF [ {
2287       \__stex_terms_invoke_op:nw { #1 }
2288     }{
2289       \__stex_terms_invoke_op:nw { #1 } []
2290     }
2291   }{
2292     \peek_charcode_remove:NTF * {
2293       \__stex_terms_invoke_text:n { #1 }
2294     }{
2295       \peek_charcode:NTF [ {
2296         \__stex_terms_invoke_math:nw { #1 }
2297       }{
2298         \__stex_terms_invoke_math:nw { #1 } []
2299       }
2300     }
2301   }
2302 }

```

(End definition for `__stex_terms_invoke_math:n`.)

`__stex_terms_invoke_op:nw`

```

2303 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2304   \__stex_terms_args:n { #2 }
2305   \cs_if_exist:cTF {
2306     stex_op_notation_ #1 \c_hash_str
2307     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2308   }{
2309     \csname stex_op_notation_ #1 \c_hash_str
2310     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2311   \endcsname
2312   }{
2313     % TODO throw error
2314   }
2315 }

```

(End definition for `__stex_terms_invoke_op:nw`.)

`__stex_terms_invoke_math:nw`

```

2316 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2317   \__stex_terms_args:n { #2 }
2318   \prop_set_eq:Nc \l_tmpa_prop {
2319     g_stex_symdecl_ #1 _prop

```

```

2320 }
2321 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2322 \seq_if_empty:NTF \l_tmpa_seq {
2323   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2324 } {
2325   \seq_if_in:NxTF \l_tmpa_seq
2326   { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2327     \use:c{
2328       stex_notation_ #1 \c_hash_str
2329       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2330       _cs
2331     }
2332   }{
2333     \str_if_empty:NTF \l__stex_terms_variant_str {
2334       \str_if_empty:NTF \l__stex_terms_lang_str {
2335         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2336         \use:c{
2337           stex_notation_ #1 \c_hash_str \l_tmpa_str
2338           _cs
2339         }
2340       }{
2341         \msg_error:nn{stex}{error/nonotation}{#1}{
2342           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2343         }
2344       }
2345     }{
2346       \msg_error:nn{stex}{error/nonotation}{#1}{
2347         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2348       }
2349     }
2350   }
2351 }
2352 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2353 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2354   \peek_charcode_remove:NTF ! {
2355     \stex_term_custom:nn { #1 } { }
2356   }{
2357     \prop_set_eq:Nc \l_tmpa_prop {
2358       g_stex_symdecl_ #1 _prop
2359     }
2360     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2361     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2362   }
2363 }

```

(End definition for `__stex_terms_invoke_text:n`.)

24.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2364 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2365 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2366 \int_new:N \l__stex_terms_downprec
2367 \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
2368 \tl_set:Nn \l__stex_terms_left_bracket_str (
2369 \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
2370 \cs_new_protected:Nn \__stex_terms_maybe_brackets:nn {
2371 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2372 \bool_if:NTF \l_stex_inarray_bool { #2 }{
2373 \dobrackets { #2 }
2374 }
2375 }{ #2 }
2376 }

(End definition for \__stex_terms_maybe_brackets:nn.)

```

```

\dobrackets
2377 %\RequirePackage{scalerel}
2378 \cs_new_protected:Npn \dobrackets #1 {
2379 %\ThisStyle{\if D@m@switch
2380 % \exp_args:Nnx \use:nn
2381 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2382 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2383 % \else
2384 \exp_args:Nnx \use:nn
2385 { \l__stex_terms_left_bracket_str #1 }
2386 { \l__stex_terms_right_bracket_str }
2387 %\fi}
2388 }

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

```

```

\withbrackets
2389 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2390 \exp_args:Nnx \use:nn
2391 {
2392 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2393 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2394 #3
2395 }
2396 {
2397 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2398 {\l__stex_terms_left_bracket_str}
2399 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2400     {\l__stex_terms_right_bracket_str}
2401   }
2402 }

```

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

`\STEXinvisible`

```

2403 \cs_new_protected:Npn \STEXinvisible #1 {
2404   \stex_annotate_invisible:n { #1 }
2405 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2406 \cs_new_protected:Nn \_stex_term_oms:nnn {
2407   \stex_annotate:nnn{ OMID }{ #2 }{
2408     \stex_highlight_term:nn { #1 } { #3 }
2409   }
2410 }
2411
2412 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2413   \__stex_terms_maybe_brackets:nn { #3 }{
2414     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2415   }
2416 }

```

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

`_stex_term_math_oma:nnnn`

```

2417 \cs_new_protected:Nn \_stex_term_oma:nnn {
2418   \stex_annotate:nnn{ OMA }{ #2 }{
2419     \stex_highlight_term:nn { #1 } { #3 }
2420   }
2421 }
2422
2423 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2424   \__stex_terms_maybe_brackets:nn { #3 }{
2425     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2426   }
2427 }

```

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

`_stex_term_math_omb:nnnn`

```

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

```

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

`_stex_term_math_arg:nnn`

```

2439 \cs_new_protected:Nn \_stex_term_arg:nn {
2440   \stex_unhighlight_term:n {
2441     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2442   }
2443 }
2444 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2445   \exp_args:Nnx \use:nn
2446     { \int_set:Nn \l__stex_terms_downprec { #2 }
2447       \_stex_term_arg:nn { #1 }{ #3 }
2448     }
2449   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2450 }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2451 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2452   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2453   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2454     \tl_set:Nn \l_tmpa_tl { #4 }
2455   }{
2456     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2457     \seq_reverse:N \l_tmpa_seq
2458     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2459     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2460
2461     \seq_map_inline:Nn \l_tmpa_seq {
2462       \exp_args:Nno \tl_set:No \l_tmpa_tl {
2463         \exp_args:Nno
2464           \l_tmpa_cs { ##1 } \l_tmpa_tl
2465       }
2466     }
2467
2468   }
2469   \exp_args:Nnno
2470   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2471 }

```

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

`\stex_term_custom:nn`

```

2472 \cs_new_protected:Nn \stex_term_custom:nn {
2473   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2474   \str_set:Nn \l_tmpa_str { #2 }
2475   \tl_clear:N \l_tmpa_tl
2476   \int_zero:N \l_tmpa_int
2477   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2478   \__stex_terms_custom_loop:
2479 }

```

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

_stex_terms_custom_loop:

```

2480 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2481   \bool_set_false:N \l_tmpa_bool
2482   \bool_while_do:nn {
2483     \str_if_eq_p:ee X {
2484       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2485     }
2486   }{
2487     \int_incr:N \l_tmpa_int
2488   }
2489
2490   \peek_charcode:NTF [ {
2491     % notation/text component
2492     \_stex_terms_custom_component:w
2493   } {
2494     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2495       % all arguments read => finish
2496       \_stex_terms_custom_final:
2497     } {
2498       % arguments missing
2499       \peek_charcode_remove:NTF * {
2500         % invisible, specific argument position or both
2501         \peek_charcode:NTF [ {
2502           % visible specific argument position
2503           \_stex_terms_custom_arg:wn
2504         } {
2505           % invisible
2506           \peek_charcode_remove:NTF * {
2507             % invisible specific argument position
2508             \_stex_terms_custom_arg_inv:wn
2509           } {
2510             % invisible next argument
2511             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2512           }
2513         }
2514       } {
2515         % next normal argument
2516         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2517       }
2518     }
2519   }
2520 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

2521 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2522   \bool_set_true:N \l_tmpa_bool
2523   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2524 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2525 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2526   \str_set:Nx \l_tmpb_str {
2527     \str_item:Nn \l_tmpa_str { #1 }
2528   }
2529   \str_case:VnTF \l_tmpb_str {
2530     { X } {
2531       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2532     }
2533     { i } { \__stex_terms_custom_set_X:n { #1 } }
2534     { b } { \__stex_terms_custom_set_X:n { #1 } }
2535     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2536     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2537   }{}{
2538     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2539   }
2540
2541   \bool_if:nTF \l_tmpa_bool {
2542     \tl_put_right:Nx \l_tmpa_tl {
2543       \stex_annotate_invisible:n {
2544         \stex_term_arg:nn { \int_eval:n { #1 } }
2545         \exp_not:n { { #2 } }
2546       }
2547     }
2548   } {
2549     \tl_put_right:Nx \l_tmpa_tl {
2550       \stex_term_arg:nn { \int_eval:n { #1 } }
2551       \exp_not:n { { #2 } }
2552     }
2553   }
2554
2555   \__stex_terms_custom_loop:
2556 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2557 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2558   \str_set:Nx \l_tmpa_str {
2559     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2560     X
2561     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2562   }
2563 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2564 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2565   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2566   \__stex_terms_custom_loop:
2567 }

```

(End definition for __stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

2568 \cs_new_protected:Nn \_stex_terms_custom_final: {
2569   \int_compare:nNnTF \l_tmpb_int = 0 {
2570     \exp_args:Nnno \_stex_term_oms:nnn
2571   }{
2572     \str_if_in:NnTF \l_tmpa_str {b} {
2573       \exp_args:Nnno \_stex_term_ombind:nnn
2574     } {
2575       \exp_args:Nnno \_stex_term_oma:nnn
2576     }
2577   }
2578   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2579 }

```

(End definition for `_stex_terms_custom_final:.`)

`\symref`

`\symname`

```

2580 \NewDocumentCommand \symref { m m }{
2581   \let\compemph_uri_prev:\compemph@uri
2582   \let\compemph@uri\symrefemph@uri
2583   \STEXsymbol{#1}![#2]
2584   \let\compemph@uri\compemph_uri_prev:
2585 }
2586
2587 \keys_define:nn { stex / symname } {
2588   post      .str_set_x:N      = \l_stex_symname_post_str
2589 }
2590
2591 \cs_new_protected:Nn \stex_symname_args:n {
2592   \str_clear:N \l_stex_symname_post_str
2593   \keys_set:nn { stex / symname } { #1 }
2594 }
2595
2596 \NewDocumentCommand \symname { 0{} m }{
2597   \stex_symname_args:n { #1 }
2598   \stex_get_symbol:n { #2 }
2599   \str_set:Nx \l_tmpa_str {
2600     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2601   }
2602   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2603
2604   \let\compemph_uri_prev:\compemph@uri
2605   \let\compemph@uri\symrefemph@uri
2606   \exp_args:NNx \use:nn
2607   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2608     \l_tmpa_str \l_stex_symname_post_str
2609   ] }
2610   \let\compemph@uri\compemph_uri_prev:
2611 }

```

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

24.3 Notation Components

2612 `<@@=stex_notationcomps>`

`\stex_highlight_term:nn`

```

2613
2614 \str_new:N \l__stex_notationcomps_highlight_uri_str
2615 \cs_new_protected:Nn \stex_highlight_term:nn {
2616   \exp_args:Nnx
2617   \use:nn {
2618     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2619     #2
2620   } {
2621     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2622     { \l__stex_notationcomps_highlight_uri_str }
2623   }
2624 }
2625
2626 \cs_new_protected:Nn \stex_unhighlight_term:n {
2627   % \latexml_if:TF {
2628   %   #1
2629   % } {
2630   %   \scalatex_if:TF {
2631   %     #1
2632   %   } {
2633     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2634   % }
2635 % }
2636 }
```

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

```

\comp
\compemph@uri 2637 \cs_new_protected:Npn \comp #1 {
\compemph      2638   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph      2639     \scalatex_if:TF {
\defemph@uri  2640       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph   2641     }{
\symrefemph@uri 2642       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2643     }
2644   }
2645 }
2646
2647 \cs_new_protected:Npn \compemph@uri #1 #2 {
2648   \compemph{ #1 }
2649 }
2650
2651
2652 \cs_new_protected:Npn \compemph #1 {
2653   \textcolor{blue}{#1}
2654 }
2655
2656 \cs_new_protected:Npn \defemph@uri #1 #2 {
2657   \defemph{#1}
2658 }
```

```

2659
2660 \cs_new_protected:Npn \defemph #1 {
2661     \textbf{#1}
2662 }
2663
2664 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2665     \symrefemph{#1}
2666 }
2667
2668 \cs_new_protected:Npn \symrefemph #1 {
2669     \textbf{#1}
2670 }

```

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

\ellipses

```

2671 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2672 \bool_new:N \l_stex_inarray_bool
2673 \bool_set_false:N \l_stex_inarray_bool
2674 \NewDocumentCommand \parray { m m } {
2675     \begingroup
2676     \bool_set_true:N \l_stex_inarray_bool
2677     \begin{array}{#1}
2678         #2
2679     \end{array}
2680 \endgroup
2681 }
2682
2683 \NewDocumentCommand \prmatrix { m } {
2684     \begingroup
2685     \bool_set_true:N \l_stex_inarray_bool
2686     \begin{matrix}
2687         #1
2688     \end{matrix}
2689 \endgroup
2690 }
2691
2692 \def \parrayline #1 #2 {
2693     #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2694 }
2695
2696 \def \parraylineh #1 #2 {
2697     #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2698 }
2699
2700 \def \parraycell #1 {
2701     #1 \bool_if:NT \l_stex_inarray_bool {&}
2702 }

```

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

```

2703 \endpackage

```

Chapter 25

STEX -Structural Features Implementation

```
2704 <*package>
2705
2706 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2707
2708 <@@=stex_features>
      Warnings and error messages
2709
```

25.1 The feature environment

structural@feature

```
2710
2711 \NewDocumentEnvironment{structural@feature}{ m m m }{
2712   \stex_if_in_module:F {
2713     \msg_set:nnn{stex}{error/nomodule}{
2714       Structural~Feature~has~to~occur~in~a~module:\\
2715       Feature~#2~of~type~#1\\
2716       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2717     }
2718     \msg_error:nn{stex}{error/nomodule}
2719   }
2720
2721   \str_set:Nx \l_stex_module_name_str {
2722     \prop_item:Nn \l_stex_current_module_prop
2723       { name } / #2 - feature
2724   }
2725
2726   \str_set:Nx \l_stex_module_ns_str {
2727     \prop_item:Nn \l_stex_current_module_prop
2728       { ns }
2729   }
2730
```

```

2731
2732 \str_clear:N \l_tmpa_str
2733 \seq_clear:N \l_tmpa_seq
2734 \tl_clear:N \l_tmpa_tl
2735 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2736   origname = #2,
2737   name      = \l_stex_module_name_str ,
2738   ns        = \l_stex_module_ns_str ,
2739   imports   = \exp_not:o { \l_tmpa_seq } ,
2740   constants = \exp_not:o { \l_tmpa_seq } ,
2741   content   = \exp_not:o { \l_tmpa_tl } ,
2742   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2743   lang      = \l_stex_module_lang_str ,
2744   sig       = \l_tmpa_str ,
2745   meta      = \l_tmpa_str ,
2746   feature   = #1 ,
2747 }
2748
2749 \stex_if_smsmode:TF {
2750   \stex_smsmode_set_codes:
2751 } {
2752   \begin{stex_annotate_env}{ feature:#1 }{}
2753   \stex_annotate_invisible:nnn{header}{}{ #3 }
2754 }
2755 }{
2756   \str_set:Nx \l_tmpa_str {
2757     c_stex_feature_
2758     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2759     \prop_item:Nn \l_stex_current_module_prop { name }
2760     _prop
2761   }
2762   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2763   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2764   \stex_if_smsmode:TF {
2765     \exp_args:Nx \stex_add_to_sms:n {
2766       \prop_gset_from_keyval:cn {
2767         c_stex_feature_
2768         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2769         \prop_item:Nn \l_stex_current_module_prop { name }
2770         _prop
2771       } {
2772         origname = #2,
2773         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2774         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2775         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2776         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2777         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2778         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2779         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2780         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2781         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2782         feature   = \prop_item:cn { \l_tmpa_str } { feature }
2783       }
2784     }

```

```

2785 } {
2786     \end{stex_annotate_env}
2787 }
2788 }
2789

```

25.2 Features

structure

```

2790
2791 \prop_new:N \l_stex_all_structures_prop
2792
2793 \keys_define:nn { stex / features / structure } {
2794     name .str_set_x:N = \l__stex_features_structure_name_str ,
2795 }
2796
2797 \cs_new_protected:Nn \__stex_features_structure_args:n {
2798     \str_clear:N \l__stex_features_structure_name_str
2799     \keys_set:nn { stex / features / structure } { #1 }
2800 }
2801
2802 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2803 % \__stex_features_structure_args:n { ##1 }
2804 % \str_if_empty:NT \l__stex_features_structure_name_str {
2805 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2806 % }
2807 %} {
2808 %
2809 %}
2810
2811 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2812     \__stex_features_structure_args:n { #1 }
2813     \str_if_empty:NT \l__stex_features_structure_name_str {
2814         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2815     }
2816     \exp_args:Nnnx
2817     \begin{structural@feature}{ structure }
2818         { \l__stex_features_structure_name_str }{}
2819         \seq_clear:N \l_tmpa_seq
2820         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2821     }{
2822         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2823         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2824         \str_set:Nx \l_tmpa_str {
2825             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2826             \prop_item:Nn \l_stex_current_module_prop { name }
2827         }
2828         \seq_map_inline:Nn \l_tmpa_seq {
2829             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2830         }
2831         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2832         \exp_args:Nnx

```

```

2834 \AddToHookNext { env / mathstructure / after }{
2835 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2836 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2837 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2838 \STEXexport {
2839 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2840 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2841 {\l_tmpa_str}
2842 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2843 {#2}{\l_tmpa_str}
2844 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2845 % \prop_item:Nn \l_stex_current_module_prop { origname },
2846 % \l_tmpa_str
2847 % }
2848 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2849 % #2,\l_tmpa_str
2850 % }
2851 % \tl_set:cx { #2 } {
2852 % \stex_invoke_structure:n { \l_tmpa_str }
2853 % }
2854 % }
2855
2856 \end{structural@feature}
2857 % \g_stex_last_feature_prop
2858 }

\instantiate

2859 \seq_new:N \l__stex_features_structure_field_seq
2860 \str_new:N \l__stex_features_structure_field_str
2861 \str_new:N \l__stex_features_structure_def_tl
2862 \prop_new:N \l__stex_features_structure_prop
2863 \NewDocumentCommand \instantiate { m O{} m }{
2864 \stex_smsmode_set_codes:
2865 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2866 \prop_set_eq:Nc \l__stex_features_structure_prop {
2867 c_stex_feature_\l_tmpa_str _prop
2868 }
2869 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2870 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2871 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2872 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2873 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2874 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2875 {!} \l_tmpa_tl
2876 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2877 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2878 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2879 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2880 }{
2881 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2882 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2883 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2884 \l_tmpa_tl
2885 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2886         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2887         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2888     }{
2889         \tl_clear:N \l_tmpb_tl
2890     }
2891 }
2892 }{
2893     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2894     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2895         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2896         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2897         \tl_clear:N \l_tmpa_tl
2898     }{
2899         % TODO throw error
2900     }
2901 }
2902 % \l_tmpa_str: name
2903 % \l_tmpa_tl: definiens
2904 % \l_tmpb_tl: notation
2905 \tl_if_empty:NT \l__stex_features_structure_field_str {
2906     % TODO throw error
2907 }
2908 \str_clear:N \l_tmpb_str
2909
2910 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2911 \seq_map_inline:Nn \l_tmpa_seq {
2912     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2913     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2914     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2915         \seq_map_break:n {
2916             \str_set:Nn \l_tmpb_str { ####1 }
2917         }
2918     }
2919 }
2920 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2921     \l_tmpb_str
2922
2923 \tl_if_empty:NTF \l_tmpb_tl {
2924     \tl_if_empty:NF \l_tmpa_tl {
2925         \exp_args:Nx \use:n {
2926             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fe
2927         }
2928     }
2929 }{
2930     \tl_if_empty:NTF \l_tmpa_tl {
2931         \exp_args:Nx \use:n {
2932             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2933         }
2934     }
2935 }{
2936     \exp_args:Nx \use:n {
2937         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2938         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2939     }

```

```

2940     }
2941   }
2942   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2943   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2944   % #3/\l_stex_features_structure_field_str
2945   % \par
2946   % \expandafter\present\csname
2947   %   g_stex_symdecl_
2948   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2949   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2950   %   #3/\l_stex_features_structure_field_str
2951   %   _prop
2952   % \endcsname
2953 }
2954
2955 \tl_clear:N \l__stex_features_structure_def_tl
2956
2957 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2958 \seq_map_inline:Nn \l_tmpa_seq {
2959   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2960   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2961   \exp_args:Nx \use:n {
2962     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2963
2964     }
2965   }
2966
2967   \prop_if_exist:cF {
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_tmpa_str
2972     _prop
2973   }{
2974     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2975     \l_tmpb_str
2976     \exp_args:Nx \use:n {
2977       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2978     }
2979   }
2980 }
2981
2982 \symdecl*[type={\STEXsymbol{module-type}}{
2983   \_stex_term_math_oms:nnnn {
2984     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2985     \prop_item:Nn \l__stex_features_structure_prop {name}
2986     }{}{0}{}
2987   }{}{#3}
2988 }
2989 % TODO: -> sms file
2990
2991 \tl_set:cx{ #3 }{
2992   \stex_invoke_structure:nnn {
2993     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```



```

2994     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2995   } {
2996     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2997     \prop_item:Nn \l__stex_features_structure_prop {name}
2998   }
2999 }
3000
3001 }

```

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

\stex_invoke_structure:nnn

```

3002 % #1: URI of the instance
3003 % #2: URI of the instantiated module
3004 \cs_new_protected:Nn \stex_invoke_structure:nnn {
3005   \tl_if_empty:nTF{ #3 }{
3006     \prop_set_eq:Nc \l__stex_features_structure_prop {
3007       c_stex_feature_ #2 _prop
3008     }
3009     \tl_clear:N \l_tmpa_tl
3010     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3011     \seq_map_inline:Nn \l_tmpa_seq {
3012       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3013       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3014       \cs_if_exist:cT {
3015         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3016       }{
3017         \tl_if_empty:NF \l_tmpa_tl {
3018           \tl_put_right:Nn \l_tmpa_tl {,}
3019         }
3020         \tl_put_right:Nx \l_tmpa_tl {
3021           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3022         }
3023       }
3024     }
3025     \exp_args:No \mathstrut \l_tmpa_tl
3026   }{
3027     \stex_invoke_symbol:n{#1/#3}
3028   }
3029 }

```

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

```

3030 </package>

```

STEX

-Statements Implementation

132

```

3063 \seq_new:N \g_stex_statements_patched_seq
3064
3065 \cs_new_protected:Nn \stex_statements_set_patched:n {
3066   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3067 }
3068
3069 \cs_new_protected:Nn \stex_statements_patch:nn {
3070   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
3071     \AddToHook{begindocument}{
3072       \cs_if_exist:cTF{end#1}{
3073         \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3074         \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3075       }{
3076         \NewDocumentEnvironment{#1}{0{}}{
3077           \use:c{__stex_statements_#2_begin:n}{ }
3078         }{
3079           \use:c{__stex_statements_#2_end:}
3080         }
3081       }
3082     }
3083   }
3084 }

```

26.1 Definitions

definition

```

3085
3086 \NewDocumentCommand \definiendum { 0{ } m m } {
3087   \stex_get_symbol:n { #2 }
3088   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3089   \scalatex_if:TF {
3090     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3091   } {
3092     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3093   }
3094 }
3095 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3096 \NewDocumentCommand \definame { 0{ } m } {
3097   % TODO: root
3098   \stex_get_symbol:n { #2 }
3099   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3100   \str_set:Nx \l_tmpa_str {
3101     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3102   }
3103   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3104   \scalatex_if:TF {
3105     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3106       \l_tmpa_str
3107     }
3108   } {
3109     \defemph@uri {
3110       \l_tmpa_str
3111     } { \l_stex_get_symbol_uri_str }

```

```

3112 }
3113 }
3114 \stex_deactivate_macro:Nn \definame {definition-environments}
3115
3116 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3117   \stex_reactivate_macro:N \definiendum
3118   \stex_reactivate_macro:N \definame
3119   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3120   \seq_clear:N \l_tmpb_seq
3121   \seq_map_inline:Nn \l_tmpa_seq {
3122     \str_if_eq:nnF{ ##1 }{}{
3123       \stex_get_symbol:n { ##1 }
3124       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3125         \l_stex_get_symbol_uri_str
3126       }
3127     }
3128   }
3129   \stex_smsmode_set_codes:
3130   \exp_args:Nnnx
3131   \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3132 }
3133
3134 \cs_new_protected:Nn \__stex_statements_defi_end: {
3135   \end{stex_annotate_env}
3136 }

```

Hook:

```

3137 \stex_statements_patch:nn{definition}{defi}

```

26.2 Assertions

assertion

```

3138 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3139   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3140   \seq_clear:N \l_tmpb_seq
3141   \seq_map_inline:Nn \l_tmpa_seq {
3142     \str_if_eq:nnF{ ##1 }{}{
3143       \stex_get_symbol:n { ##1 }
3144       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3145         \l_stex_get_symbol_uri_str
3146       }
3147     }
3148   }
3149   \titleemph{Assertion}~
3150   \stex_smsmode_set_codes:
3151   \exp_args:Nnnx
3152   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3153 }
3154
3155 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3156   \end{stex_annotate_env}
3157 }

```

Hook:

```
3158 \stex_statements_patch:nn{assertion}{assertion}
```

theorem

```
3159 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3160   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3161   \seq_clear:N \l_tmpb_seq
3162   \seq_map_inline:Nn \l_tmpa_seq {
3163     \str_if_eq:nnF{ ##1 }{}{
3164       \stex_get_symbol:n { ##1 }
3165       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3166         \l_stex_get_symbol_uri_str
3167       }
3168     }
3169   }
3170   \titleemph{Theorem}~
3171   \stex_smsmode_set_codes:
3172   \exp_args:Nnnx
3173   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3174 }
3175
3176 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3177   \end{stex_annotate_env}
3178 }
```

Hook:

```
3179 \stex_statements_patch:nn{theorem}{theorem}
```

lemma

```
3180 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3181   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3182   \seq_clear:N \l_tmpb_seq
3183   \seq_map_inline:Nn \l_tmpa_seq {
3184     \str_if_eq:nnF{ ##1 }{}{
3185       \stex_get_symbol:n { ##1 }
3186       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3187         \l_stex_get_symbol_uri_str
3188       }
3189     }
3190   }
3191   \titleemph{Lemma}~
3192   \stex_smsmode_set_codes:
3193   \exp_args:Nnnx
3194   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3195 }
3196
3197 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3198   \end{stex_annotate_env}
3199 }
```

Hook:

```
3200 \stex_statements_patch:nn{lemma}{lemma}
```

axiom

```
3201 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3202   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3203   \seq_clear:N \l_tmpb_seq
3204   \seq_map_inline:Nn \l_tmpa_seq {
3205     \str_if_eq:nnF{ ##1 }{}{
3206       \stex_get_symbol:n { ##1 }
3207       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3208         \l_stex_get_symbol_uri_str
3209       }
3210     }
3211   }
3212   \titleemph{Axiom}~
3213   \stex_smsmode_set_codes:
3214   \exp_args:Nnnx
3215   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3216 }
3217
3218 \cs_new_protected:Nn \__stex_statements_axiom_end: {
3219   \end{stex_annotate_env}
3220 }

```

Hook:

```
3221 \stex_statements_patch:nn{axiom}{axiom}
```

26.3 Examples

example

```
3222 \cs_new_protected:Nn \__stex_statements_example_begin:n {
3223   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3224   \seq_clear:N \l_tmpb_seq
3225   \seq_map_inline:Nn \l_tmpa_seq {
3226     \str_if_eq:nnF{ ##1 }{}{
3227       \stex_get_symbol:n { ##1 }
3228       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3229         \l_stex_get_symbol_uri_str
3230       }
3231     }
3232   }
3233   \titleemph{Example}~
3234   \stex_smsmode_set_codes:
3235   \exp_args:Nnnx
3236   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3237 }
3238
3239 \cs_new_protected:Nn \__stex_statements_example_end: {
3240   \end{stex_annotate_env}
3241 }

```

Hook:

```
3242 \stex_statements_patch:nn{example}{example}
```

26.4 OMText

```

3243 \keys_define:nn { stex / omtex } {
3244   id      .str_set_x:N    = \l_stex_omtext_id_str ,
3245   title   .tl_set:N       = \l_stex_omtext_title_tl ,
3246   type    .tl_set_x:N     = \l_stex_omtext_type_tl ,
3247   for     .tl_set_x:N     = \l_stex_omtext_for_tl ,
3248   from    .tl_set_x:N     = \l_stex_omtext_from_tl ,
3249   start   .tl_set:N       = \l_stex_omtext_start_tl ,
3250 }
3251 \cs_new_protected:Nn \stex_omtext_args:n {
3252   \tl_clear:N \l_stex_omtext_title_tl
3253   \tl_clear:N \l_stex_omtext_start_tl
3254   \keys_set:nn { stex / omtex } { #1 }
3255 }
3256 \newif\if@in@omtext\@in@omtextfalse
3257 \NewDocumentEnvironment {omtext} { 0{ } } {
3258   \stex_omtext_args:n { #1 }
3259   \tl_if_empty:NTF \l_stex_omtext_start_tl {
3260     \tl_if_empty:NF \l_stex_omtext_title_tl {
3261       \titleemph{\l_stex_omtext_title_tl}:~
3262     }
3263   }{
3264     \titleemph{\l_stex_omtext_start_tl}~
3265   }
3266   \@in@omtexttrue
3267
3268   \stex_ref_new_doc_target:n \l_stex_omtext_id_str
3269   \stex_smsmode_set_codes:
3270   \ignorespacesandpars
3271 }{}
3272 \</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

```
3273 <*package>
3274 <@@=stex_sproof>
3275
3276 %%%%%%%%%%% sproof.dtx %%%%%%%%%%%
3277
```

27.2 Proofs

We first define some keys for the proof environment.

```
3278 \keys_define:nn { stex / spf } {
3279   id          .str_set:N = \l__stex_sproof_spf_id_str,
3280   display     .tl_set:N  = \l__stex_sproof_spf_display_tl,
3281   for         .tl_set:N  = \l__stex_sproof_spf_for_tl ,
3282   from        .tl_set:N  = \l__stex_sproof_spf_from_tl ,
3283   proofend    .tl_set:N  = \l__stex_sproof_spf_proofend_tl,
3284   type        .tl_set:N  = \l__stex_sproof_spf_type_tl,
3285   title       .tl_set:N  = \l__stex_sproof_spf_title_tl,
3286   continues   .tl_set:N  = \l__stex_sproof_spf_continues_tl,
3287   functions   .tl_set:N  = \l__stex_sproof_spf_functions_tl,
3288   method      .tl_set:N  = \l__stex_sproof_spf_method_tl
3289 }
3290 \cs_new_protected:Nn \__stex_sproof_spf_args:n {
3291   \str_clear:N \l__stex_sproof_spf_id_str
3292   \tl_clear:N \l__stex_sproof_spf_display_tl
3293   \tl_clear:N \l__stex_sproof_spf_for_tl
3294   \tl_clear:N \l__stex_sproof_spf_from_tl
3295   \tl_set:Nn \l__stex_sproof_spf_proofend_tl {\sproof@box}
3296   \tl_clear:N \l__stex_sproof_spf_type_tl
3297   \tl_clear:N \l__stex_sproof_spf_title_tl

```

¹⁰EDNOTE: need an implementation for L^AT_EX_ML


```

3298 \tl_clear:N \l__stex_sproof_spf_continues_tl
3299 \tl_clear:N \l__stex_sproof_spf_functions_tl
3300 \tl_clear:N \l__stex_sproof_spf_method_tl
3301 \keys_set:nn { stex / spf }{ #1 }
3302 }

```

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

```

3303 \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.

```

3304 \newcount\count_ten
3305 \newenvironment{pst@with@label}[1]{
3306   \edef\pst@label{#1}
3307   \advance\count_ten by 1\relax
3308   \count_ten=1
3309 }{
3310   \advance\count_ten by -1\relax
3311 }

```

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

```

3312 \def\the@pst@label{
3313   \pst@make@label\pst@label{\number\count_ten}\l__stex_sproof_pstlabel_postfix_tl
3314 }

```

(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.

```

3315 \keys_define:nn { stex / pstlabel }{
3316   prefix      .tl_set:N   = \l__stex_sproof_pstlabel_prefix_tl,
3317   delimiter   .tl_set:N   = \l__stex_sproof_pstlabel_delimiter_tl,
3318   postfix     .tl_set:N   = \l__stex_sproof_pstlabel_postfix_tl
3319 }
3320 \cs_new_protected:Nn \__stex_sproof_pstlabel_args:n {

```

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

```

3321 \tl_set:Nn \l__stex_sproof_pstlabel_prefix_tl {P}
3322 \tl_set:Nn \l__stex_sproof_pstlabel_delimiter_tl {.}
3323 \tl_clear:N \l__stex_sproof_pstlabel_postfix_tl
3324 }
3325 \__stex_sproof_pstlabel_args:n {}
3326 \newrobustcmd\setpstlabelstyle[1]{
3327   \__stex_sproof_pstlabel_args:n {#1}
3328 }
3329 \newrobustcmd\setpstlabelstyledefault{%
3330   \__stex_sproof_pstlabel_args:n{prefix=P,delimiter=.,postfix={}}
3331 }

```

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

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

```

3332 \ExplSyntaxOff
3333 \def\pst@make@label@long#1#2{\@for\@I:=#1\do{\expandafter\expandafter\expandafter\@I\csname
3334 \def\pst@make@label@angles#1#2{\ensuremath{\@for\@I:=#1\do{\rangle}}#2}
3335 \def\pst@make@label@short#1#2{#2}
3336 \def\pst@make@label@empty#1#2{}
3337 \ExplSyntaxOn
3338 \def\pstlabelstyle#1{%
3339   \def\pst@make@label{\use:c{pst@make@label@#1}}%
3340 }%
3341 \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.

```

3342 \def\next@pst@label{%
3343   \global\advance\count\count10 by 1%
3344 }%

```

(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

```

3345 \def\sproof@box{
3346   \hbox{\vrule\vbox{\hrule width 6 pt\vskip 6pt\hrule}\vrule}
3347 }
3348 \def\spf@proofend{\sproof@box}
3349 \def\sproofend{
3350   \tl_if_empty:NF \l__stex_sproof_spf_proofend_tl {
3351     \hfil\null\nobreak\hfill\l__stex_sproof_spf_proofend_tl\par\smallskip
3352   }
3353 }
3354 \def\sProofEndSymbol#1{\def\sproof@box{#1}}

```

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

spf@*@kw

```

3355 \def\spf@proofsketch@kw{Proof Sketch}
3356 \def\spf@proof@kw{Proof}
3357 \def\spf@step@kw{Step}

```

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

For the other languages, we set up triggers

```

3358
3359 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3360 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3361   \input{sproof-ngerman.ldf}
3362 }
3363 \clist_if_in:NnT \l_tmpa_clist {finnish}{
3364   \input{sproof-finnish.ldf}
3365 }
3366 \clist_if_in:NnT \l_tmpa_clist {french}{
3367   \input{sproof-french.ldf}
3368 }
3369 \clist_if_in:NnT \l_tmpa_clist {russian}{
3370   \input{sproof-russian.ldf}
3371 }
3372

```

spfsketch

```

3373 \newrobustcmd\spfsketch[2][]{
3374   \__stex_sproof_spf_args:n{#1}
3375   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3376     \titleemph{
3377       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3378         \spf@proofsketch@kw
3379       }{
3380         \l__stex_sproof_spf_type_tl
3381       }
3382     }:
3383   }
3384   {-#2}
3385   %\sref@label@id{this \ifx\spf@type\@empty\spf@proofsketch@kw\else\spf@type\fi}
3386   \sproofend
3387 }

```

(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

```

3388 \newenvironment{spfeq}[2][]{
3389   \__stex_sproof_spf_args:n{#1}
3390   %\sref@target
3391   \tl_if_eq:NnF \l__stex_sproof_spf_display_tl\spf@flow{
3392     \titleemph{
3393       \tl_if_empty:NtF \l__stex_sproof_spf_type_tl {
3394         \spf@proof@kw
3395       }{
3396         \l__stex_sproof_spf_type_tl
3397       }
3398     }:
3399   }

```

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

¹²EDNOTE: document above

```

3400   {~#2}
3401   \begin{displaymath}\begin{array}{rcll}
3402   }{
3403   \end{array}\end{displaymath}
3404   }

```

(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.

```

3405 \newenvironment{spf@proof}[2] []{
3406   \__stex_sproof_spf_args:n{#1}
3407   %\sref@target
3408   \count_ten=10
3409   \par\noindent
3410   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3411     \titleemph{
3412       \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {
3413         \spf@proof@kw
3414       }{
3415         \l__stex_sproof_spf_type_tl
3416       }
3417     }:
3418   }
3419   {~#2}
3420   %\sref@label@id{this \ifx\spf@type\@empty\spf@proof@kw\else\spf@type\fi}
3421   \def\pst@label{}
3422   \newcount\pst@count% initialize the labeling mechanism
3423   \begin{description}\begin{pst@with@label}{\l__stex_sproof_pstlabel_prefix_tl}
3424   }{
3425     \end{pst@with@label}\end{description}
3426   }
3427   \newenvironment{sproof}[2] []{\begin{spf@proof}[#1]{#2}}{\sproofend\end{spf@proof}}
3428   \newenvironment{sProof}[2] []{\begin{spf@proof}[#1]{#2}}{\end{spf@proof}}

```

\spfidea

```

3429 \newrobustcmd\spfidea[2] []{
3430   \__stex_sproof_spf_args:n{#1}
3431   \titleemph{
3432     \tl_if_empty:NTF \l__stex_sproof_spf_type_tl {Proof~Idea}{
3433       \l__stex_sproof_spf_type_tl
3434     }:
3435   }~#2
3436   \sproofend
3437 }

```

(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

```

3438 \newenvironment{spfstep}[1][]{
3439   \_stex_sproof_spf_args:n{#1}
3440   \@in@omtexttrue
3441   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3442     \item[\the@pst@label]
3443   }
3444   \tl_if_empty:NF \l__stex_sproof_spf_title_tl {
3445     {(\titleemph{\l__stex_sproof_spf_title_tl})\enspace}
3446   }
3447   %\sref@label@id{\pst@label}
3448   \ignorespacesandpars
3449 }{
3450   \next@pst@label\ignorespacesandpars
3451 }

```

sproofcomment

```

3452 \newenvironment{sproofcomment}[1][]{
3453   \_stex_sproof_spf_args:n{#1}
3454   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3455     \item[\the@pst@label]
3456   }
3457 }{
3458   \next@pst@label
3459 }

```

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.

```

3460 \newenvironment{subproof}[2][]{
3461   \_stex_sproof_spf_args:n{#1}
3462   \def\@test{#2}
3463   \ifx\@test\empty\else
3464     \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3465       \item[\the@pst@label]
3466     }{#2}
3467   \fi
3468   \begin{pst@with@label}{\pst@label,\number\count_ten}
3469 }{
3470   \end{pst@with@label}\next@pst@label
3471 }

```

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

```

3472 \newenvironment{spfcases}[2][]{
3473   \def\@test{#1}
3474   \ifx\@test\empty
3475     \begin{subproof}[method=by-cases]{#2}
3476   \else
3477     \begin{subproof}[#1,method=by-cases]{#2}
3478   \fi
3479 }{

```

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

```

3480 \end{subproof}
3481 }

```

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

```

3482 \newenvironment{spfcase}[2] [] {
3483   \__stex_sproof_spf_args:n{#1}
3484   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3485     \item[\the@pst@label]
3486   }
3487   \def\@test{#2}
3488   \ifx\@test\@empty
3489   \else
3490     {\titleemph{#2}:~}
3491   \fi
3492   \begin{pst@with@label}{\pst@label,\number\count_ten}
3493 }{
3494   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3495     \sproofend
3496   }
3497   \end{pst@with@label}
3498   \next@pst@label
3499 }

```

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

```

3500 \newrobustcmd\spfcasesketch[3] [] {
3501   \__stex_sproof_spf_args:n{#1}
3502   \tl_if_eq:NNF \l__stex_sproof_spf_display_tl\spf@flow{
3503     \item[\the@pst@label]
3504   }
3505   \def\@test{#2}
3506   \ifx\@test\@empty
3507   \else
3508     {\titleemph{#2}:~}
3509   \fi#3
3510   \next@pst@label
3511 }%

```

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.

```

3512 \keys_define:nn { stex / just }{
3513   id          .str_set:x:N = \l__stex_sproof_just_id_str,
3514   method      .tl_set:N    = \l__stex_sproof_just_method_tl,
3515   premises    .tl_set:N    = \l__stex_sproof_just_premises_tl,
3516   args        .tl_set:N    = \l__stex_sproof_just_args_tl
3517 }

```

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

```
3518 \newenvironment{justification}[1] [] {}{}
```

\premise

```
3519 \newrobustcmd\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.

```
3520 \newrobustcmd\justarg[2] [] {#2}
```

```
3521 \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

```
3522 <*package>
3523
3524 %%%%%%%%%% others.dtx %%%%%%%%%%
3525
3526 <@@=stex_others>
    Warnings and error messages
3527 % None

\MSC Math subject classifier

3528 \NewDocumentCommand \MSC {m} {
3529 % TODO
3530 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3531 \@ifpackageloaded{tikzinput}{
3532 \RequirePackage{stex-tikzinput}
3533 }{}
3534 </package>
```


Chapter 29

STEX -Metatheory Implementation

```
3535 <*package>
3536 <@@=stex_modules>
3537
3538 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3539
3540 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3541 \begingroup
3542 \stex_module_setup:nn{
3543   ns=\c_stex_metatheory_ns_str,
3544   meta=NONE
3545 }{Metatheory}
3546 \stex_reactivate_macro:N \symdecl
3547 \stex_reactivate_macro:N \notation
3548 \stex_reactivate_macro:N \symdef
3549 \ExplSyntaxOff
3550 \csname stex_suppress_html:n\endcsname{
3551   % is-a (a:A, a \in A, a is an A, etc.)
3552   \symdecl[args=ai]{isa}
3553   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3554   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3555   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3556
3557   % bind (\forall, \Pi, \lambda etc.)
3558   \symdecl[args=Bi]{bind}
3559   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3560   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3561   \notation[deffun]{bind}{\comp( #1 \comp{ }\;\to\; )}{#1 \comp, #2}
3562
3563   % dummy variable
3564   \symdecl{dummyvar}
3565   \notation[underscore]{dummyvar}{\comp\_}
3566   \notation[dot]{dummyvar}{\comp\cdot}
3567   \notation[dash]{dummyvar}{\comp{\rm --}}
3568
3569   %fromto (function space, Hom-set, implication etc.)
```

```

3570 \symdecl[args=ai]{fromto}
3571 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3572 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3573
3574 % mapto (lambda etc.)
3575 %\symdecl[args=Bi]{mapto}
3576 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3577 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3578 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3579
3580 % function/operator application
3581 \symdecl[args=ia]{apply}
3582 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3583 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3584
3585 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3586 \symdecl{collection}
3587 \notation[U]{collection}{\comp{\mathcal{U}}}
3588 \notation[set]{collection}{\comp{\textsf{Set}}}
3589
3590 % sequences
3591 \symdecl[args=1]{seqtype}
3592 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3593
3594 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3595 \notation[ui]{sequence-index}{#1^{#2}}
3596
3597 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3598 %\notation[ui]{sequence-from-to}{#1^{#2}\comp{\,\ellipses,}#1^{#3}}
3599 % ^ superceded by \aseqfromto and \livar/\uivar
3600
3601 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3602 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses,}#2}{#1\comp,#2}
3603 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses,}#2\comp{\,\ellipses,}#3}
3604
3605 % letin (‘‘let’’, local definitions, variable substitution)
3606 \symdecl[args=bii]{letin}
3607 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3608 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3609 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3610
3611 % structures
3612 \symdecl*[args=1]{module-type}
3613 \notation{module-type}{\mathtt{MOD} #1}
3614 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3615 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3616
3617 }
3618 \ExplSyntaxOn
3619 \stex_add_to_current_module:n{
3620   \let\nappa\apply
3621   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3622   \def\livar{\csname sequence-index\endcsname[li]}
3623   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3624     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3625     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3626     \def\nappe#1#2#3{\apply{#1}{\aseqfromto{#2}{#3}}}
3627   }
3628   \__stex_modules_end_module:
3629   \endgroup
3630 \endpackage

```

Chapter 30

Tikzinput Implementation

```
3631 <*package>
3632
3633 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3634
3635 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3636 \RequirePackage{l3keys2e}
3637
3638 \keys_define:nn { tikzinput } {
3639   image .bool_set:N = \c_tikzinput_image_bool,
3640   image .default:n = false ,
3641 }
3642
3643 \ProcessKeysOptions { tikzinput }
3644
3645 \bool_if:NTF \c_tikzinput_image_bool {
3646   \RequirePackage{graphicx}
3647
3648   \providecommand\usetikzlibrary[]{}
3649   \newcommand\tikzinput[2] [] {\includegraphics[#1]{#2}}
3650 }{
3651   \RequirePackage{tikz}
3652   \RequirePackage{standalone}
3653
3654   \newcommand \tikzinput [2] [] {
3655     \setkeys{Gin}{#1}
3656     \ifx \Gin@ewidth \Gin@exclamation
3657       \ifx \Gin@eheight \Gin@exclamation
3658         \input { #2 }
3659       \else
3660         \resizebox{!}{ \Gin@eheight }{
3661           \input { #2 }
3662         }
3663       \fi
3664     \else
3665       \ifx \Gin@eheight \Gin@exclamation
3666         \resizebox{ \Gin@ewidth }{!}{
3667           \input { #2 }
3668         }
3669       \fi
3670     \fi
3671   }
3672 }
```

```

3669         \else
3670         \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3671             \input { #2 }
3672         }
3673         \fi
3674     \fi
3675 }
3676 }
3677
3678 \newcommand \ctikzinput [2] [] {
3679     \begin{center}
3680         \tikzinput [1] {#2}
3681     \end{center}
3682 }
3683
3684 \@ifpackageloaded{stex}{
3685     \RequirePackage{stex-tikzinput}
3686 }{}
3687
3688 \</package>
3689 \<*stex>
3690 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3691 \RequirePackage{stex}
3692 \RequirePackage{tikzinput}
3693
3694 \newcommand\mhtikzinput[2] [] {%
3695     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3696     \stex_in_repository:nn\Gin@mhrepos{
3697         \tikzinput[#1]{\mhp\path{##1}{#2}}
3698     }
3699 }
3700 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
3701 \</stex>

```

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

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.

```
3702 \*cls)
3703 \<@@=document_structure>
3704 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3705 \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`

```
3706 \keys_define:nn{ document-structure / pkg }{
3707   class      .str_set_x:N = \c_document_structure_class_str,
3708   minimal    .bool_set:N = \c_document_structure_minimal_bool,
3709   report     .code:n      = {
3710     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3711     \str_set:Nn \c_document_structure_class_str {report}
3712   },
3713   book       .code:n      = {
3714     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3715     \str_set:Nn \c_document_structure_class_str {book}
3716   },
3717   bookpart   .code:n      = {
3718     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3719     \str_set:Nn \c_document_structure_class_str {book}
3720     \str_set:Nn \c_document_structure_topsect_str {chapter}
3721   },
```

```

3722 docopt      .str_set_x:N = \c_document_structure_docopt_str,
3723 unknown     .code:n      = {
3724   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3725 }
3726 }
3727 \ProcessKeysOptions{ document-structure / pkg }
3728 \str_if_empty:NT \c_document_structure_class_str {
3729   \str_set:Nn \c_document_structure_class_str {article}
3730 }
3731 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_docopt_str]
3732   {\c_document_structure_class_str}
3733

```

31.3 Beefing up the document environment

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

```

3734 \RequirePackage{omdoc}
3735 \bool_if:NF \c_document_structure_minimal_bool {
3736   \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

```

3737 \keys_define:nn { document-structure / document }{
3738   id .str_set_x:N = \c_document_structure_document_id_str
3739 }
3740 \let\__document_structure_orig_document=\document
3741 \renewcommand{\document}[1][]{
3742   \keys_set:nn{ document-structure / document }{ #1 }
3743   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3744   \__document_structure_orig_document
3745 }

```

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

```

3746 }
3747 \</cls>

```

31.4 Implementation: OMDoc Package

```

3748 \*package>
3749 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3750 \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

```

3751
3752 \keys_define:nn{ document-structure / pkg }{
3753   class      .str_set_x:N = \c_document_structure_class_str,
3754   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3755   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3756 }
3757 \ProcessKeysOptions{ document-structure / pkg }
3758 \str_if_empty:NT \c_document_structure_class_str {
3759   \str_set:Nn \c_document_structure_class_str {article}
3760 }
3761 \str_if_empty:NT \c_document_structure_topsect_str {
3762   \str_set:Nn \c_document_structure_topsect_str {section}
3763 }

```

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

```

3764 \RequirePackage{xspace}
3765 \RequirePackage{comment}
3766 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3767 \@ifpackageloaded{babel}{
3768   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3769   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3770     \input{omdoc-ngerman.ldf}
3771   }
3772 }{}
3773 %\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.

```

3774 \int_new:N \l_document_structure_section_level_int
3775 \str_case:VnF \c_document_structure_topsect_str {
3776   {part}{
3777     \int_set:Nn \l_document_structure_section_level_int {0}
3778   }
3779   {chapter}{
3780     \int_set:Nn \l_document_structure_section_level_int {1}
3781   }
3782 }{
3783   \str_case:VnF \c_document_structure_class_str {
3784     {book}{
3785       \int_set:Nn \l_document_structure_section_level_int {0}
3786     }
3787     {report}{
3788       \int_set:Nn \l_document_structure_section_level_int {0}
3789     }
3790   }{
3791     \int_set:Nn \l_document_structure_section_level_int {2}
3792   }
3793 }

```


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

```
3794 \def\current@section@level{document}%
3795 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3796 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3797 \cs_new_protected:Npn \skipomgroup {
3798   \ifcase\l_document_structure_section_level_int
3799   \or\stepcounter{part}
3800   \or\stepcounter{chapter}
3801   \or\stepcounter{section}
3802   \or\stepcounter{subsection}
3803   \or\stepcounter{subsubsection}
3804   \or\stepcounter{paragraph}
3805   \or\stepcounter{subparagraph}
3806   \fi
3807 }
```

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

`blindomgroup`

```
3808 \newcommand\at@begin@blindomgroup[1]{%
3809 \newenvironment{blindomgroup}
3810 {
3811   \int_incr:N\l_document_structure_section_level_int
3812   \at@begin@blindomgroup\l_document_structure_section_level_int
3813 }{}}
```

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

```
3814 \newcommand\omgroup@nonum[2]{
3815   \ifx\hyper@anchor\@undefined\else\phantomsection\fi
3816   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3817 }
```

(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.

```
3818 \newcommand\omgroup@num[2]{
```

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

```

3819 \tl_if_empty:NTF \l__document_structure_omgroup_short_tl {
3820   \@nameuse{#1}{#2}
3821 }{
3822   \cs_if_exist:NTF\rdfmata@sectioning{
3823     \@nameuse{rdfmata@#1@old}[\l__document_structure_omgroup_short_tl]{#2}
3824   }{
3825     \@nameuse{#1}[\l__document_structure_omgroup_short_tl]{#2}
3826   }
3827 }
3828 %\sref@label@id@arg{\omdoc@ssect@name~\@nameuse{the#1}}\omgroup@id
3829 }

```

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

omgroup

```

3830 \keys_define:nn { document-structure / omgroupp }{
3831   id          .str_set_x:N = \l__document_structure_omgroup_id_str,
3832   date        .str_set_x:N = \l__document_structure_omgroup_date_str,
3833   creators    .clist_set:N = \l__document_structure_omgroup_creators_clist,
3834   contributors .clist_set:N = \l__document_structure_omgroup_contributors_clist,
3835   srccite     .tl_set:N    = \l__document_structure_omgroup_srccite_tl,
3836   type        .tl_set:N    = \l__document_structure_omgroup_type_tl,
3837   short       .tl_set:N    = \l__document_structure_omgroup_short_tl,
3838   display     .tl_set:N    = \l__document_structure_omgroup_display_tl,
3839   intro       .tl_set:N    = \l__document_structure_omgroup_intro_tl,
3840   loadmodules .bool_set:N  = \l__document_structure_omgroup_loadmodules_bool
3841 }
3842 \cs_new_protected:Nn \l__document_structure_omgroup_args:n {
3843   \str_clear:N \l__document_structure_omgroup_id_str
3844   \str_clear:N \l__document_structure_omgroup_date_str
3845   \clist_clear:N \l__document_structure_omgroup_creators_clist
3846   \clist_clear:N \l__document_structure_omgroup_contributors_clist
3847   \tl_clear:N \l__document_structure_omgroup_srccite_tl
3848   \tl_clear:N \l__document_structure_omgroup_type_tl
3849   \tl_clear:N \l__document_structure_omgroup_short_tl
3850   \tl_clear:N \l__document_structure_omgroup_display_tl
3851   \tl_clear:N \l__document_structure_omgroup_intro_tl
3852   \bool_set_false:N \l__document_structure_omgroup_loadmodules_bool
3853   \keys_set:nn { document-structure / omgroupp } { #1 }
3854 }

```

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 omgroupp, i.e. after the section heading.

```

3855 \newif\if@mainmatter\@mainmattertrue
3856 \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.

```

3857 \keys_define:nn { document-structure / sectioning }{
3858   name .str_set_x:N = \l__document_structure_sect_name_str ,
3859   ref .str_set_x:N = \l__document_structure_sect_ref_str ,
3860   clear .bool_set:N = \l__document_structure_sect_clear_bool ,
3861   num .bool_set:N = \l__document_structure_sect_num_bool ,
3862 }

```

```

3863 \cs_new_protected:Nn \__document_structure_sect_args:n {
3864   \str_clear:N \l__document_structure_sect_name_str
3865   \str_clear:N \l__document_structure_sect_ref_str
3866   \bool_set_false:N \l__document_structure_sect_clear_bool
3867   \bool_set_false:N \l__document_structure_sect_num_bool
3868   \keys_set:nn { document-structure / sectioning } { #1 }
3869 }
3870 \newcommand\omdoc@sectioning[3][]{
3871   \__document_structure_sect_args:n {#1}
3872   \let\omdoc@sect@name\l__document_structure_sect_name_str
3873   \bool_if:NT \l__document_structure_sect_clear_bool { \cleardoublepage }
3874   \if@mainmatter% numbering not overridden by frontmatter, etc.
3875     \bool_if:NTF \l__document_structure_sect_num_bool {
3876       \omgroup@num{#2}{#3}
3877     }{
3878       \omgroup@nonum{#2}{#3}
3879     }
3880     \def\current@section@level{\omdoc@sect@name}
3881   \else
3882     \omgroup@nonum{#2}{#3}
3883   \fi
3884 }% 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.

```

3885 \newcommand\omgroup@redefine@addtocontents[1]{%
3886   %\edef\__document_structureimport{#1}%
3887   %\@for\@I:=\__document_structureimport\do{%
3888     %\edef\@path{\csname module@\@I @path\endcsname}%
3889     %\@ifundefined{tf@toc}\relax%
3890     %    {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3891   %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3892   %\def\addcontentsline##1##2##3{%
3893     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3894   %\else% hyperref.sty not loaded
3895   %\def\addcontentsline##1##2##3{%
3896     %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3897   %\fi
3898 }% 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.

```

3899 \int_new:N \l_document_structure_omgroup_level_int
3900 \newenvironment{omgroup}[2][]{% keys, title
3901 {
3902   \__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.

```

3903 \bool_if:NT \l__document_structure_omgroup_loadmodules_bool {
3904   \omgroup@redefine@addtocontents{
3905     %\@ifundefined{module@id}\used@modules%
3906     %{\@ifundefined{module@\module@id @path}{\used@modules}\module@id}

```

```

3907     }
3908 }

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

3909 \int_incr:N \l_document_structure_omgroup_level_int
3910 \int_incr:N \l_document_structure_section_level_int
3911 \ifcase\l_document_structure_section_level_int
3912   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3913   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3914   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3915   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3916   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3917   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3918   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3919 \fi
3920 \at@begin@omgroup[#1]\l_document_structure_section_level_int{#2}
3921 \stex_ref_new_doc_target:n\l__document_structure_omgroup_id_str
3922 }% for customization
3923 {}

```

and finally, we localize the sections

```

3924 \newcommand\omdoc@part@kw{Part}
3925 \newcommand\omdoc@chapter@kw{Chapter}
3926 \newcommand\omdoc@section@kw{Section}
3927 \newcommand\omdoc@subsection@kw{Subsection}
3928 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3929 \newcommand\omdoc@paragraph@kw{paragraph}
3930 \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`

```

3931 \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).

```

3932 \cs_if_exist:NTF\frontmatter{
3933   \let\__document_structure_orig_frontmatter\frontmatter
3934   \let\frontmatter\relax
3935 }{
3936   \tl_set:Nn\__document_structure_orig_frontmatter{
3937     \clearpage
3938     \@mainmatterfalse
3939     \pagenumbering{roman}
3940   }
3941 }
3942 \cs_if_exist:NTF\backmatter{

```

```

3943 \let\__document_structure_orig_backmatter\backmatter
3944 \let\backmatter\relax
3945 }{
3946 \tl_set:Nn\__document_structure_orig_backmatter{
3947 \clearpage
3948 \@mainmatterfalse
3949 \pagenumbering{roman}
3950 }
3951 }

```

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.

```

3952 \newenvironment{frontmatter}{
3953 \__document_structure_orig_frontmatter
3954 }{
3955 \cs_if_exist:NTF\mainmatter{
3956 \mainmatter
3957 }{
3958 \clearpage
3959 \@mainmattertrue
3960 \pagenumbering{arabic}
3961 }
3962 }

```

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

```

3963 \newenvironment{backmatter}{
3964 \__document_structure_orig_backmatter
3965 }{
3966 \cs_if_exist:NTF\mainmatter{
3967 \mainmatter
3968 }{
3969 \clearpage
3970 \@mainmattertrue
3971 \pagenumbering{arabic}
3972 }
3973 }

```

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

```

3974 \@mainmattertrue\pagenumbering{arabic}

```

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

```

3975 \newcommand\afterprematurestop{}
3976 \def\prematurestop@endomgroup{
3977 \int_compare:nNf \l_document_structure_omgroup_level_int = 0 {
3978 \end{omgroup}
3979 \int_decr:N \l_document_structure_omgroup_level_int
3980 \prematurestop@endomgroup
3981 }
3982 }
3983 \providecommand\prematurestop{

```

```

3984 \message{Stopping sTeX processing prematurely}
3985 \prematurestop@endomgroup
3986 \afterprematurestop
3987 \end{document}
3988 }

```

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

31.8 Global Variables

\setSGvar set a global variable

```

3989 \RequirePackage{etoolbox}
3990 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

3991 \newrobustcmd\useSGvar[1]{%
3992   \@ifundefined{sTeX@Gvar@#1}
3993   {\PackageError{omdoc}
3994    {The sTeX Global variable #1 is undefined}
3995    {set it with \protect\setSGvar}}
3996   \@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.

```

3997 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
3998   \@ifundefined{sTeX@Gvar@#1}
3999   {\PackageError{omdoc}
4000    {The sTeX Global variable #1 is undefined}
4001    {set it with \protect\setSGvar}}
4002   {\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.

```
4003 \*cls)
4004 \@@=mikoslides)
4005 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
4006 \RequirePackage{l3keys2e,expl-keystr-compatible}
4007
4008 \keys_define:nn{mikoslides / cls}{
4009   class .code:n = {
4010     \PassOptionsToClass{\CurrentOption}{omdoc}
4011     \str_if_eq:nnT{#1}{book}{
4012       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4013     }
4014     \str_if_eq:nnT{#1}{report}{
4015       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
4016     }
4017   },
4018   notes .bool_set:N = \c__mikoslides_notes_bool ,
4019   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
4020   unknown .code:n = {
4021     \PassOptionsToClass{\CurrentOption}{omdoc}
4022     \PassOptionsToClass{\CurrentOption}{beamer}
4023     \PassOptionsToPackage{\CurrentOption}{mikoslides}
4024   }
4025 }
4026 \ProcessKeysOptions{ mikoslides / cls }
4027 \bool_if:NTF \c__mikoslides_notes_bool {
4028   \PassOptionsToPackage{notes=true}{mikoslides}
4029 }{
4030   \PassOptionsToPackage{notes=false}{mikoslides}
4031 }
4032 \</cls)
```

now we do the same for the mikoslides package.

```

4033 <*package>
4034 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
4035 \RequirePackage{l3keys2e,expl-keystr-compat}
4036
4037 \keys_define:nn{mikoslides / pkg}{
4038   topsect      .str_set_x:N = \c__mikoslides_topsect_str,
4039   defaulttopsect .str_set_x:N = \c__mikoslides_defaulttopsec_str,
4040   notes        .bool_set:N = \c__mikoslides_notes_bool ,
4041   slides        .code:n      = { \bool_set_false:N \c__mikoslides_notes_bool },
4042   sectocframes .bool_set:N = \c__mikoslides_sectocframes_bool ,
4043   frameimages .bool_set:N = \c__mikoslides_frameimages_bool ,
4044   fiboxed      .bool_set:N = \c__mikoslides_fiboxed_bool ,
4045   nopproblems .bool_set:N = \c__mikoslides_nopproblems_bool,
4046   unknown      .code:n      = {
4047     \PassOptionsToClass{\CurrentOption}{stex}
4048     \PassOptionsToClass{\CurrentOption}{tikzinput}
4049   }
4050 }
4051 \ProcessKeysOptions{ mikoslides / pkg }
4052 \newif\ifnotes
4053 \bool_if:NTF \c__mikoslides_notes_bool {
4054   \notesttrue
4055 }{
4056   \notesfalse
4057 }
4058

```

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

```

4059 \str_if_empty:NTF \c__mikoslides_topsect_str {
4060   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
4061 }{
4062   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
4063 }
4064 </package>

```

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

```

4065 <*cls>
4066 \bool_if:NTF \c__mikoslides_notes_bool {
4067   \LoadClass{omdoc}
4068 }{
4069   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
4070   \newcounter{Item}
4071   \newcounter{paragraph}
4072   \newcounter{subparagraph}
4073   \newcounter{Hfootnote}
4074   \RequirePackage{omdoc}
4075 }

```

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

```

4076 \RequirePackage{mikoslides}
4077 </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.

```

4078 \*package>
4079 \RequirePackage{stex-compatibility}
4080 \RequirePackage{stex-tikzinput}
4081 \bool_if:NT \c__mikoslides_notes_bool {
4082   \RequirePackage{a4wide}
4083   \RequirePackage{marginnote}
4084   \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
4085   \RequirePackage{mdframed}
4086   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
4087   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
4088 }
4089 \RequirePackage{etoolbox}
4090 \RequirePackage{amssymb}
4091 \RequirePackage{amsmath}
4092 \RequirePackage{comment}
4093 \RequirePackage{textcomp}
4094 \RequirePackage{url}
4095 \RequirePackage{graphicx}
4096 \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

```

4097 \bool_if:NT \c__mikoslides_notes_bool {
4098   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
4099 }

```

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

```

4100 \newcounter{slide}
4101 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
4102 \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.

```

4103 \bool_if:NTF \c__mikoslides_notes_bool {
4104   \renewenvironment{note}{\ignorespaces}{\}
4105 }{
4106   \excludecomment{note}
4107 }

```

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.

```
4108 \bool_if:NT \c__mikoslides_notes_bool {
4109   \newlength{\slideframewidth}
4110   \setlength{\slideframewidth}{1.5pt}
```

frame We first define the keys.

```
4111 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
4112   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
4113     \bool_set_true:N #1
4114   }{
4115     \bool_set_false:N #1
4116   }
4117 }
4118 \keys_define:nn{mikoslides / frame}{
4119   label .str_set_x:N = \l__mikoslides_frame_label_str,
4120   allowframebreaks .code:n = {
4121     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
4122   },
4123   allowdisplaybreaks .code:n = {
4124     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
4125   },
4126   fragile .code:n = {
4127     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
4128   },
4129   shrink .code:n = {
4130     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
4131   },
4132   squeeze .code:n = {
4133     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
4134   },
4135   t .code:n = {
4136     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
4137   },
4138 }
4139 \cs_new_protected:Nn \__mikoslides_frame_args:n {
4140   \str_clear:N \l__mikoslides_frame_label_str
4141   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
4142   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
4143   \bool_set_true:N \l__mikoslides_frame_fragile_bool
4144   \bool_set_true:N \l__mikoslides_frame_shrink_bool
4145   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
4146   \bool_set_true:N \l__mikoslides_frame_t_bool
4147   \keys_set:nn { mikoslides / frame }{ #1 }
4148 }
```

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

```
4149 \renewenvironment{frame}[1][]{
4150   \__mikoslides_frame_args:n{#1}
4151   \sffamily
4152   \stepcounter{slide}
4153   \def\@currentlabel{\theslide}
4154   \str_if_empty:NF \l__mikoslides_frame_label_str {
4155     \label{\l__mikoslides_frame_label_str}
```

4156 }

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

```

4157 \def\itemize@level{outer}
4158 \def\itemize@outer{outer}
4159 \def\itemize@inner{inner}
4160 \renewcommand\newpage{\addtocounter{framenum}{1}}
4161 \newcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
4162 \renewenvironment{itemize}{
4163   \ifx\itemize@level\itemize@outer
4164     \def\itemize@label{\$ \rhd \$}
4165   \fi
4166   \ifx\itemize@level\itemize@inner
4167     \def\itemize@label{\$ \scriptstyle \rhd \$}
4168   \fi
4169   \begin{list}
4170     {\itemize@label}
4171     {\setlength{\labelsep}{.3em}
4172      \setlength{\labelwidth}{.5em}
4173      \setlength{\leftmargin}{1.5em}
4174     }
4175   \edef\itemize@level{\itemize@inner}
4176 }{
4177   \end{list}
4178 }

```

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

```

4179 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
4180 }{
4181   \medskip\miko@slidelabel\end{mdframed}
4182 }

```

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

`\frametitle`

```

4183 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
4184 }

```

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

EdN:18

`\pause` 18

```

4185 \bool_if:NT \c__mikoslides_notes_bool {
4186   \newcommand\pause{}
4187 }

```

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

`nomtext`

```

4188 \bool_if:NTF \c__mikoslides_notes_bool {
4189   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}
4190 }{
4191   \excludecomment{nomtext}
4192 }

```

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

nomgroup

```
4193 \bool_if:NTF \c__mikoslides_notes_bool {
4194   \newenvironment{nomgroup}[2] [] {\begin{omgroup}[#1]{#2}}{\end{omgroup}}
4195 }{
4196   \excludecomment{nomgroup}
4197 }
```

ndefinition

```
4198 \bool_if:NTF \c__mikoslides_notes_bool {
4199   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
4200 }{
4201   \excludecomment{ndefinition}
4202 }
```

nassertion

```
4203 \bool_if:NTF \c__mikoslides_notes_bool {
4204   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
4205 }{
4206   \excludecomment{nassertion}
4207 }
```

nsproof

```
4208 \bool_if:NTF \c__mikoslides_notes_bool {
4209   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
4210 }{
4211   \excludecomment{nsproof}
4212 }
```

nexample

```
4213 \bool_if:NTF \c__mikoslides_notes_bool {
4214   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
4215 }{
4216   \excludecomment{nexample}
4217 }
```

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

```
4218 \def\inputref@preskip{\smallskip}
4219 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
4220 \let\orig@inputref\inputref
4221 \def\inputref{\@ifstar\ninputref\orig@inputref}
4222 \newcommand\ninputref[2] [] {
4223   \bool_if:NT \c__mikoslides_notes_bool {
4224     \orig@inputref[#1]{#2}
4225   }
4226 }
```

(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>}`.

```

4227 \newlength{\slidelogoheight}
4228
4229 \bool_if:NTF \c__mikoslides_notes_bool {
4230   \setlength{\slidelogoheight}{.4cm}
4231 }{
4232   \setlength{\slidelogoheight}{1cm}
4233 }
4234 \newsavebox{\slidelogo}
4235 \sbox{\slidelogo}{\TeX}
4236 \newrobustcmd{\setslidelogo}[1]{
4237   \sbox{\slidelogo}{\includegraphics[height=\slidelogoheight]{#1}}
4238 }
```

(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.

```

4239 \def\source{Michael Kohlhase}% customize locally
4240 \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.

```

4241 \def\copyrightnotice{\footnotesize\copyright : \hspace{.3ex}{\source}}
4242 \newsavebox{\cclogo}
4243 \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{cc_somerights}}
4244 \newif\ifcchref\cchreffalse
4245 \AtBeginDocument{
4246   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
4247 }
4248 \def\licensing{
4249   \ifcchref
4250     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
4251   \else
4252     {\usebox{\cclogo}}
4253   \fi
4254 }
4255 \newrobustcmd{\setlicensing}[2][]{
4256   \def@url{#1}
4257   \sbox{\cclogo}{\includegraphics[height=\slidelogoheight]{#2}}
4258   \ifx@url@empty
4259     \def\licensing{{\usebox{\cclogo}}}
4260   \else
4261     \def\licensing{
```

```

4262     \ifcchref
4263     \href{#1}{\usebox{\cclogo}}
4264     \else
4265     {\usebox{\cclogo}}
4266     \fi
4267   }
4268 \fi
4269 }

```

(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

```

4270 \newrobustcmd\miko@slidelabel{
4271   \vbox to \slidelogoheight{
4272     \vss\hbox to \slidewidth
4273     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
4274   }
4275 }

```

(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.

```

4276 \def\Gin@mhrepos{}
4277 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
4278 \define@key{Gin}{label}{\def\@currentlabel{\arabic{slide}}\label{#1}}
4279 \newrobustcmd\frameimage[2][]{
4280   \stepcounter{slide}
4281   \bool_if:NT \c__mikoslides_frameimages_bool {
4282     \def\Gin@ewidth{}\setkeys{Gin}{#1}
4283     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4284     \begin{center}
4285       \bool_if:NTF \c__mikoslides_fiboxed_bool {
4286         \fbox{
4287           \ifx\Gin@ewidth\@empty
4288             \ifx\Gin@mhrepos\@empty
4289               \mhgraphics[width=\slidewidth,#1]{#2}
4290             \else
4291               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4292             \fi
4293           \else% Gin@ewidth empty
4294             \ifx\Gin@mhrepos\@empty
4295               \mhgraphics[#1]{#2}
4296             \else
4297               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4298             \fi
4299           \fi% Gin@ewidth empty
4300         }
4301       }{
4302         \ifx\Gin@ewidth\@empty

```

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

```

4303         \ifx\Gin@mhrepos\@empty
4304             \mhgraphics[width=\slidewidth,#1]{#2}
4305         \else
4306             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4307         \fi
4308         \ifx\Gin@mhrepos\@empty
4309             \mhgraphics[#1]{#2}
4310         \else
4311             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4312         \fi
4313     \fi% Gin@ewidth empty
4314 }
4315 \end{center}
4316 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
4317 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4318 }
4319 } % 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.

```

4320 \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

```

4321 \AddToHook{begindocument}{
4322     \definecolor{green}{rgb}{0,.5,0}
4323     \definecolor{purple}{cmyk}{.3,1,0,.17}
4324 }

```

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`.

```

4325 % \def\STpresent#1{\textcolor{blue}{#1}}
4326 \def\defemph#1{\textcolor{magenta}{#1}}
4327 \def\symrefemph#1{\textcolor{cyan}{#1}}
4328 \def\compemph#1{\textcolor{blue}{#1}}
4329 \def\titleemph#1{\textcolor{blue}{#1}}
4330 \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.

```

4331 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4332 \def\smalltextwarning{
4333     \pgfuseimage{miko@small@dbend}
4334     \xspace
4335 }
4336 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}

```

```

4337 \newrobustcmd\textwarning{
4338   \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4339   \xspace
4340 }
4341 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}
4342 \newrobustcmd\bigtextwarning{
4343   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4344   \xspace
4345 }
(End definition for \textwarning. This function is documented on page ??.)
4346 \newrobustcmd\putgraphicsat[3]{
4347   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4348 }
4349 \newrobustcmd\putat[2]{
4350   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4351 }

```

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`.

```

4352 \bool_if:NT \c__mikoslides_sectocframes_bool {
4353   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4354     \newcounter{chapter}\counterwithin*{section}{chapter}
4355   }{
4356     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4357       \newcounter{chapter}\counterwithin*{section}{chapter}
4358     }
4359   }
4360 }

```

`\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
4361 \@ifpackageloaded{omdoc}{
4362   \str_case:VnF \__mikoslidestopsect {
4363     {part}{
4364       \int_set:Nn \l_document_structure_section_level_int {0}
4365       \def\thesection{\arabic{chapter}.\arabic{section}}
4366       \def\part@prefix{\arabic{chapter}.}
4367     }
4368     {chapter}{
4369       \int_set:Nn \l_document_structure_section_level_int {1}
4370       \def\thesection{\arabic{chapter}.\arabic{section}}
4371       \def\part@prefix{\arabic{chapter}.}
4372     }
4373   }{
4374     \int_set:Nn \l_document_structure_section_level_int {2}
4375     \def\part@prefix{}
4376   }

```



```

4377 }
4378
4379 \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 chooses the L^AT_EX sectioning macros according to `\section@level`.

omgroup

```

4380 \renewenvironment{omgroup}[2][]{
4381   \_document_structure_omgroup_args:n { #1 }
4382   \int_incr:N \l_document_structure_omgroup_level_int
4383   \int_incr:N \l_document_structure_section_level_int
4384   \bool_if:NT \c__mikoslides_sectocframes_bool {
4385     \stepcounter{slide}
4386     \begin{frame}[noframenumbering]
4387       \vfill\Large\centering
4388       \red{
4389         \ifcase\l_document_structure_section_level_int\or
4390           \stepcounter{part}
4391           \def\_mikoslideslabel{\omdoc@part@kw~\Roman{part}}
4392           \def\currentsectionlevel{\omdoc@part@kw}
4393         \or
4394           \stepcounter{chapter}
4395           \def\_mikoslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4396           \def\currentsectionlevel{\omdoc@chapter@kw}
4397         \or
4398           \stepcounter{section}
4399           \def\_mikoslideslabel{\part@prefix\arabic{section}}
4400           \def\currentsectionlevel{\omdoc@section@kw}
4401         \or
4402           \stepcounter{subsection}
4403           \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4404           \def\currentsectionlevel{\omdoc@subsection@kw}
4405         \or
4406           \stepcounter{subsubsection}
4407           \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4408           \def\currentsectionlevel{\omdoc@subsubsection@kw}
4409         \or
4410           \stepcounter{mparagraph}
4411           \def\_mikoslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}.\arabic{mparagraph}}
4412           \def\currentsectionlevel{\omdoc@paragraph@kw}
4413         \fi% end ifcase
4414         \_mikoslideslabel\sref@label@id\_mikoslideslabel
4415         \quad #2%
4416       }%
4417       \vfill%
4418     \end{frame}%
4419   }
4420   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4421 }{}
4422 }

```

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

```

4423 \def\inserttheorembodyfont{\normalfont}
4424 \bool_if:NF \c__mikoslides_notes_bool {
4425   \defbeamertemplate{theorem begin}{miko}
4426   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4427     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%
4428     \inserttheorempunctuation\inserttheorembodyfont\xspace}
4429   \defbeamertemplate{theorem end}{miko}{}}

```

and we set it as the default one.

```

4430 \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.

```

4431 \expandafter\def\csname Parent2\endcsname{}
4432 }
4433 \bool_if:NT \c__mikoslides_notes_bool {
4434   \renewenvironment{columns}[1][]{%
4435     \par\noindent%
4436     \begin{minipage}%
4437       \slidewidth\centering\leavevmode%
4438   }{%
4439     \end{minipage}\par\noindent%
4440   }%
4441   \newsavebox\columnbox%
4442   \renewenvironment<>{column}[2][{%
4443     \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4444   }{%
4445     \end{minipage}\end{lrbox}\usebox\columnbox%
4446   }%
4447 }
4448 \bool_if:NTF \c__mikoslides_noproblems_bool {
4449   \newenvironment{problems}{}{}
4450 }{
4451   \excludecomment{problems}
4452 }

```

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.

```

4453 \gdef\printexcursions{}
4454 \newcommand\excursionref[2]{% label, text
4455   \bool_if:NT \c__mikoslides_notes_bool {
4456     \begin{omtext}[title=Excursion]
4457       #2 \sref[fallback=the appendix]{#1}.
4458     \end{omtext}
4459   }
4460 }
4461 \newcommand\activate@excursion[2][{}{
4462   \gappto\printexcursions{\inputref[#1]{#2}}
4463 }
4464 \newcommand\excursion[4][{}{ repos, label, path, text

```

```

4465 \bool_if:NT \c__mikoslides_notes_bool {
4466   \activate@excursion[#1]{#3}\excursionref{#2}{#4}
4467 }
4468 }

```

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

\excursiongroup

```

4469 \keys_define:nn{mikoslides / excursiongroup }{
4470   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4471   intro       .tl_set:N   = \l__mikoslides_excursion_intro_tl,
4472   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4473 }
4474 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4475   \tl_clear:N \l__mikoslides_excursion_intro_tl
4476   \str_clear:N \l__mikoslides_excursion_id_str
4477   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4478   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4479 }
4480 \newcommand\excursiongroup[1][]{
4481   \__mikoslides_excursion_args:n{ #1 }
4482   \ifdefempty\printexcursions{}% only if there are excursions
4483   {
4484     \begin{omgroup}[#1]{Excursions}%
4485     \ifdefempty\l__mikoslides_excursion_intro_tl{
4486       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4487         \l__mikoslides_excursion_intro_tl
4488       }
4489     }
4490     \printexcursions%
4491     \end{omgroup}
4492   }
4493 }
4494 \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.

```
4495 <*package>
4496 <@@=problems>
4497 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4498 \RequirePackage{l3keys2e,expl-keystr-compat}
4499
4500 \keys_define:nn { problem / pkg }{
4501   notes      .default:n    = { true },
4502   notes      .bool_set:N   = \c__problems_notes_bool,
4503   gnotes     .default:n    = { true },
4504   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4505   hints      .default:n    = { true },
4506   hints      .bool_set:N   = \c__problems_hints_bool,
4507   solutions  .default:n    = { true },
4508   solutions  .bool_set:N   = \c__problems_solutions_bool,
4509   pts        .default:n    = { true },
4510   pts        .bool_set:N   = \c__problems_pts_bool,
4511   min        .default:n    = { true },
4512   min        .bool_set:N   = \c__problems_min_bool,
4513   boxed      .default:n    = { true },
4514   boxed      .bool_set:N   = \c__problems_boxed_bool
4515 }
4516 \def\solutionstrue{
4517   \bool_set_true:N \c__problems_solutions_bool
4518 }
4519 \def\solutionsfalse{
4520   \bool_set_false:N \c__problems_solutions_bool
4521 }
4522
4523 \ProcessKeysOptions{ problem / pkg }
4524 \RequirePackage{stex-compatibility}
```

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

```
4525 \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.

```
4526 \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.

```
4527 \def\prob@problem@kw{Problem}
4528 \def\prob@solution@kw{Solution}
4529 \def\prob@hint@kw{Hint}
4530 \def\prob@note@kw{Note}
4531 \def\prob@gnote@kw{Grading}
4532 \def\prob@pt@kw{pt}
4533 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
4534 \@ifpackageloaded{babel}{
4535   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4536   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4537     \input{problem-ngerman.ldf}
4538   }
4539   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4540     \input{problem-finnish.ldf}
4541   }
4542   \clist_if_in:NnT \l_tmpa_clist {french}{
4543     \input{problem-french.ldf}
4544   }
4545   \clist_if_in:NnT \l_tmpa_clist {russian}{
4546     \input{problem-russian.ldf}
4547   }
4548 }{}
```

33.2 Problems and Solutions

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

```
4549 \keys_define:nn{ problem / problem }{
4550   id      .str_set:x:N = \l__problems_prob_id_str,
4551   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4552   min     .tl_set:N    = \l__problems_prob_min_tl,
4553   title   .tl_set:N    = \l__problems_prob_title_tl,
4554   refnum  .int_set:N   = \l__problems_prob_refnum_int
4555 }
4556 \cs_new_protected:Nn \__problems_prob_args:n {
4557   \str_clear:N \l__problems_prob_id_str
4558   \tl_clear:N \l__problems_prob_pts_tl
4559   \tl_clear:N \l__problems_prob_min_tl
4560   \tl_clear:N \l__problems_prob_title_tl
4561   \int_zero_new:N \l__problems_prob_refnum_int
```

```

4562 \keys_set:nn { problem / problem }{ #1 }
4563 \int_compare:nNnT \l__problems_prob_refnum_int = 0 {
4564   \let\l__problems_inclprob_refnum_int\undefined
4565 }
4566 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4567 \newcounter{problem}
4568 \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.

```

4569 \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

```

4570 \newcommand\prob@number{
4571   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4572     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4573   }{
4574     \int_if_exist:NTF \l__problems_prob_refnum_int {
4575       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4576     }{
4577       \prob@label\theproblem
4578     }
4579   }
4580 }

```

(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.

```

4581 \newcommand\prob@title[3]{%
4582   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4583     #2 \l__problems_inclprob_title_tl #3
4584   }{
4585     \tl_if_exist:NTF \l__problems_prob_title_tl {
4586       #2 \l__problems_prob_title_tl #3
4587     }{
4588       #1
4589     }
4590   }
4591 }

```

(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.

```

4592 \def\prob@heading{
4593   \prob@problem@kw~\prob@number\prob@title{~}{~}{~}\strut}
4594   %\sref@label{id{\prob@problem@kw~\prob@number}}{~}
4595 }

```

(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`

```

4596 \newenvironment{problem}[1][1]{
4597   \__problems_prob_args:n{#1}%\sref@target%
4598   \@in@omtexttrue% we are in a statement (for inline definitions)
4599   \stepcounter{problem}\record@problem
4600   \def\current@section@level{\prob@problem@kw}
4601   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4602 }%
4603   {\smallskip}
4604   \bool_if:NT \c__problems_boxed_bool {
4605     \surroundwithmdframed{problem}
4606   }

```

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

```

4607 \def\record@problem{
4608   \protected@write\@auxout{}
4609   {
4610     \string\@problem{\prob@number}
4611     {
4612       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4613         \l__problems_inclprob_pts_tl
4614       }{
4615         \l__problems_prob_pts_tl
4616       }
4617     }%
4618     {
4619       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4620         \l__problems_inclprob_min_tl
4621       }{
4622         \l__problems_prob_min_tl
4623       }
4624     }
4625   }
4626 }

```

(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).

```

4627 \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 its own keys that we need to define first.

```

4628 \keys_define:nn { problem / solution }{
4629   id          .str_set_x:N = \l__problems_solution_id_str ,
4630   for         .tl_set:N    = \l__problems_solution_for_tl ,
4631   height      .dim_set:N   = \l__problems_solution_height_dim ,
4632   creators    .clist_set:N = \l__problems_solution_creators_clist ,
4633   contributors .clist_set:N = \l__problems_solution_contributors_clist ,
4634   srccite     .tl_set:N    = \l__problems_solution_srccite_tl
4635 }
4636 \cs_new_protected:Nn \__problems_solution_args:n {
4637   \str_clear:N \l__problems_solution_id_str
4638   \tl_clear:N \l__problems_solution_for_tl
4639   \tl_clear:N \l__problems_solution_srccite_tl
4640   \clist_clear:N \l__problems_solution_creators_clist
4641   \clist_clear:N \l__problems_solution_contributors_clist
4642   \dim_zero:N \l__problems_solution_height_dim
4643   \keys_set:nn { problem / solution }{ #1 }
4644 }

```

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

```

4645 \newcommand\@startsolution[1][ ]{
4646   \__problems_solution_args:n { #1 }
4647   \@in@omtexttrue% we are in a statement.
4648   \bool_if:NF \c__problems_boxed_bool { \hrule }
4649   \smallskip\noindent
4650   {\textbf\prob@solution@kw : \enspace}
4651   \begin{small}
4652   \def\current@section@level{\prob@solution@kw}
4653   \ignorespacesandpars
4654 }

```

\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.

```

4655 \newcommand\startsolutions{
4656   \specialcomment{solution}{\@startsolution}{
4657     \bool_if:NF \c__problems_boxed_bool {
4658       \hrule\medskip
4659     }
4660     \end{small}%
4661   }
4662   \bool_if:NT \c__problems_boxed_bool {
4663     \surroundwithmdframed{solution}
4664   }
4665 }

```

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

\stopsolutions

```

4666 \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.

```

4667 \bool_if:NTF \c__problems_solutions_bool {
4668   \startsolutions
4669 }{
4670   \stopsolutions
4671 }

```

exnote

```

4672 \bool_if:NTF \c__problems_notes_bool {
4673   \newenvironment{exnote}[1][]{
4674     \par\smallskip\hrule\smallskip
4675     \noindent\textbf{\prob@note@kw : }\small
4676   }{
4677     \smallskip\hrule
4678   }
4679 }{
4680   \excludecomment{exnote}
4681 }

```

hint

```

4682 \bool_if:NTF \c__problems_notes_bool {
4683   \newenvironment{hint}[1][]{
4684     \par\smallskip\hrule\smallskip
4685     \noindent\textbf{\prob@hint@kw :~ }\small
4686   }{
4687     \smallskip\hrule
4688   }
4689   \newenvironment{exhint}[1][]{
4690     \par\smallskip\hrule\smallskip
4691     \noindent\textbf{\prob@hint@kw :~ }\small
4692   }{
4693     \smallskip\hrule
4694   }
4695 }{
4696   \excludecomment{hint}
4697   \excludecomment{exhint}
4698 }

```

gnote

```

4699 \bool_if:NTF \c__problems_notes_bool {
4700   \newenvironment{gnote}[1][]{
4701     \par\smallskip\hrule\smallskip
4702     \noindent\textbf{\prob@gnote@kw : }\small
4703   }{
4704     \smallskip\hrule
4705   }
4706 }{
4707   \excludecomment{gnote}
4708 }

```

33.3 Multiple Choice Blocks

```

4709 \newenvironment{mcb}{
4710   \begin{enumerate}
4711 }{
4712   \end{enumerate}
4713 }

```

we define the keys for the mcc macro

```

4714 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4715   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4716     \bool_set_true:N #1
4717   }{
4718     \bool_set_false:N #1
4719   }
4720 }
4721 \keys_define:nn { problem / mcc }{
4722   id          .str_set:x:N = \l__problems_mcc_id_str ,
4723   feedback    .tl_set:N    = \l__problems_mcc_feedback_tl ,
4724   T           .default:n   = { true } ,
4725   T           .bool_set:N   = \l__problems_mcc_t_bool ,
4726   F           .default:n   = { true } ,
4727   F           .bool_set:N   = \l__problems_mcc_f_bool ,
4728   Ttext       .code:n      = {
4729     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4730   } ,
4731   Ftext       .code:n      = {
4732     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4733   }
4734 }
4735 \cs_new_protected:Nn \l__problems_mcc_args:n {
4736   \str_clear:N \l__problems_mcc_id_str
4737   \tl_clear:N \l__problems_mcc_feedback_tl
4738   \bool_set_true:N \l__problems_mcc_t_bool
4739   \bool_set_true:N \l__problems_mcc_f_bool
4740   \bool_set_true:N \l__problems_mcc_Ttext_bool
4741   \bool_set_false:N \l__problems_mcc_Ftext_bool
4742   \keys_set:nn { problem / mcc }{ #1 }
4743 }

```

\mcc

```

4744 \newcommand\mcc[2][] {
4745   \l__problems_mcc_args:n{ #1 }
4746   \item #2
4747   \bool_if:NT \c__problems_solutions_bool {
4748     \
4749     \bool_if:NT \l__problems_mcc_t_bool {
4750       % TODO!
4751       % \ifcsstring{mcc@T}{T}{ }\{mcc@Ttext}%
4752     }
4753     \bool_if:NT \l__problems_mcc_f_bool {

```

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

```

4754      % TODO!
4755      % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4756    }
4757    \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4758      !
4759    }{
4760      \l__problems_mcc_feedback_tl
4761    }
4762  }
4763 } %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.

```

4764
4765 \keys_define:nn{ problem / inclproblem }{
4766   % id      .str_set_x:N = \l__problems_inclprob_id_str,
4767   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4768   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4769   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4770   refnum   .int_set:N    = \l__problems_inclprob_refnum_int,
4771   mhrepos  .str_set_x:N = \l__problems_inclprob_mhrepos_str
4772 }
4773
4774 \cs_new_protected:Nn \__problems_inclprob_args:n {
4775   % \str_clear:N \l__problems_prob_id_str
4776   \tl_clear:N \l__problems_inclprob_pts_tl
4777   \tl_clear:N \l__problems_inclprob_min_tl
4778   \tl_clear:N \l__problems_inclprob_title_tl
4779   \int_zero_new:N \l__problems_inclprob_refnum_int
4780   \str_clear:N \l__problems_inclprob_mhrepos_str
4781   \keys_set:nn { problem / inclproblem }{ #1 }
4782   \tl_if_empty:NT \l__problems_inclprob_pts_tl {
4783     \let\l__problems_inclprob_pts_tl\undefined
4784   }
4785   \tl_if_empty:NT \l__problems_inclprob_min_tl {
4786     \let\l__problems_inclprob_min_tl\undefined
4787   }
4788   \tl_if_empty:NT \l__problems_inclprob_title_tl {
4789     \let\l__problems_inclprob_title_tl\undefined
4790   }
4791   \int_compare:nNnT \l__problems_inclprob_refnum_int = 0 {
4792     \let\l__problems_inclprob_refnum_int\undefined
4793   }
4794 }
4795
4796 \cs_new_protected:Nn \__problems_inclprob_clear: {
4797   % \str_clear:N \l__problems_prob_id_str
4798   \let\l__problems_inclprob_pts_tl\undefined
4799   \let\l__problems_inclprob_min_tl\undefined

```

```

4799 \let\l__problems_inclprob_title_tl\undefined
4800 \let\l__problems_inclprob_refnum_int\undefined
4801 \let\l__problems_inclprob_mhrepos_str\undefined
4802 }
4803
4804 \newcommand\includeproblem[2][]{
4805   \__problems_inclprob_args:n{ #1 }
4806   \str_if_empty:NTF \l__problems_inclprob_mhrepos_str {
4807     \input{#2}
4808   }{
4809     \stex_in_repository:nn{\l__problems_inclprob_mhrepos_str}{
4810       \input{\mhpath{\l__problems_inclprob_mhrepos_str}{#2}}
4811     }
4812   }
4813   \__problems_inclprob_clear:
4814 }

```

(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.

```

4815 \AddToHook{enddocument}{
4816   \bool_if:NT \c__problems_pts_bool {
4817     \message{Total:~\arabic{pts}~points}
4818   }
4819   \bool_if:NT \c__problems_min_bool {
4820     \message{Total:~\arabic{min}~minutes}
4821   }
4822 }

```

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

```

4823 \def\pts#1{
4824   \bool_if:NT \c__problems_pts_bool {
4825     \marginpar{#1~\prob@pt@kw}
4826   }
4827 }
4828 \def\min#1{
4829   \bool_if:NT \c__problems_min_bool {
4830     \marginpar{#1~\prob@min@kw}
4831   }
4832 }

```

\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.

```

4833 \newcounter{pts}
4834 \def\show@pts{
4835   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4836     \bool_if:NT \c__problems_pts_bool {
4837       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4838       \addtocounter{pts}{\l__problems_inclprob_pts_tl}

```

```

4839     }
4840   }{
4841     \tl_if_exist:NT \l__problems_prob_pts_tl {
4842       \bool_if:NT \c__problems_pts_bool {
4843         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
4844         \addtocounter{pts}{\l__problems_prob_pts_tl}
4845       }
4846     }
4847   }
4848 }

```

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

\show@min

```

4849 \newcounter{min}
4850 \def\show@min{
4851   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4852     \bool_if:NT \c__problems_min_bool {
4853       \marginpar{\l__problems_inclprob_pts_tl;min}
4854       \addtocounter{min}{\l__problems_inclprob_min_tl}
4855     }
4856   }{
4857     \tl_if_exist:NT \l__problems_prob_min_tl {
4858       \bool_if:NT \c__problems_min_bool {
4859         \marginpar{\l__problems_prob_min_tl;min}
4860         \addtocounter{min}{\l__problems_prob_min_tl}
4861       }
4862     }
4863   }
4864 }
4865 \</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.

```
4866 \@@=hwexam>
4867 \*cls>
4868 \ProvidesExplClass{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4869 \RequirePackage{l3keys2e,expl-keystr-compatible}
4870 \DeclareOption*{
4871   \PassOptionsToClass{\CurrentOption}{omdoc}
4872   \PassOptionsToPackage{\CurrentOption}{stex}
4873   \PassOptionsToPackage{\CurrentOption}{hwexam}
4874   \PassOptionsToPackage{\CurrentOption}{tikzinput}
4875 }
4876 \ProcessOptions
```

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

```
4877 \LoadClass{omdoc}
4878 \RequirePackage{stex}
4879 \RequirePackage{hwexam}
4880 \RequirePackage{tikzinput}
4881 \RequirePackage{graphicx}
4882 \RequirePackage{a4wide}
4883 \RequirePackage{amssymb}
4884 \RequirePackage{amstext}
4885 \RequirePackage{amsmath}
```

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

```

4886 \newcommand\assig@default@type{\hwexam@assignment@kw}
4887 \def\document@hwexamtype{\assig@default@type}
4888 <@@=document_structure>
4889 \keys_define:nn { document-structure / document }{
4890 id .str_set_x:N = \c_document_structure_document_id_str,
4891 hwexamtype .tl_set:N = \document@hwexamtype
4892 }
4893 <@@=hwexam>
4894 </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.

```
4895 \*package>
4896 \ProvidesExplPackage{hwexam}{2019/03/20}{1.1}{homework assignments and exams}
4897 \RequirePackage{l3keys2e,expl-keystr-compat}
4898
4899 \newif\iftest\testfalse
4900 \DeclareOption{test}{\testtrue}
4901 \newif\ifmultiple\multiplefalse
4902 \DeclareOption{multiple}{\multipletrue}
4903 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{problem}}
4904 \ProcessOptions
```

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

```
4905 \RequirePackage{keyval}[1997/11/10]
4906 \RequirePackage{problem}
```

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

```
4907 \newcommand\hwexam@assignment@kw{Assignment}
4908 \newcommand\hwexam@given@kw{Given}
4909 \newcommand\hwexam@due@kw{Due}
4910 \newcommand\hwexam@testemptypage@kw{This page was intentionally left blank for extra
4911 space}%
4912 \newcommand\correction@probs@kw{prob.}%
4913 \newcommand\correction@pts@kw{total}%
4914 \newcommand\correction@reached@kw{reached}%
4915 \newcommand\correction@sum@kw{Sum}%
4916 \newcommand\correction@grade@kw{grade}%
4917 \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

```

4918 \ifpackageloaded{babel}{\RequirePackage[base]{babel}}
4919
4920 \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4921 \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4922   \input{hwexam-ngerman.ldf}
4923 }
4924 \clist_if_in:NnT \l_tmpa_clist {finnish}{
4925   \input{hwexam-finnish.ldf}
4926 }
4927 \clist_if_in:NnT \l_tmpa_clist {french}{
4928   \input{hwexam-french.ldf}
4929 }
4930 \clist_if_in:NnT \l_tmpa_clist {russian}{
4931   \input{hwexam-russian.ldf}
4932 }

```

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.

```

4933 \newcounter{assignment}
4934 \numberproblemsin{assignment}
4935 \renewcommand\prob@label[1]{\arabic{assignment}.#1}

```

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

```

4936 \keys_define:nn { hwexam / assignment } {
4937   id .str_set:N = \l__hwexam_assign_id_str,
4938   number .int_set:N = \l__hwexam_assign_number_int,
4939   title .tl_set:N = \l__hwexam_assign_title_tl,
4940   type .tl_set:N = \l__hwexam_assign_type_tl,
4941   given .tl_set:N = \l__hwexam_assign_given_tl,
4942   due .tl_set:N = \l__hwexam_assign_due_tl,
4943   loadmodules .code:n = {
4944     \bool_set_true:N \l__hwexam_assign_loadmodules_bool
4945   }
4946 }
4947 \cs_new_protected:Nn \__hwexam_assignment_args:n {
4948   \str_clear:N \l__hwexam_assign_id_str
4949   \int_set:Nn \l__hwexam_assign_number_int {-1}
4950   \tl_clear:N \l__hwexam_assign_title_tl
4951   \tl_clear:N \l__hwexam_assign_type_tl
4952   \tl_clear:N \l__hwexam_assign_given_tl
4953   \tl_clear:N \l__hwexam_assign_due_tl
4954   \bool_set_false:N \l__hwexam_assign_loadmodules_bool
4955   \keys_set:nn { hwexam / assignment }{ #1 }
4956 }

```

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.

```

4957 \newcommand\given@due[2]{
4958 \bool_lazy_all:nF {
4959 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
4960 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
4961 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
4962 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
4963 }{ #1 }
4964
4965 \tl_if_empty:NTF \l__hwexam_inclasssign_given_tl {
4966 \tl_if_empty:NF \l__hwexam_assign_given_tl {
4967 \hwexam@given@kw\xspace\l__hwexam_assign_given_tl
4968 }
4969 }{
4970 \hwexam@given@kw\xspace\l__hwexam_inclasssign_given_tl
4971 }
4972
4973 \bool_lazy_or:nnF {
4974 \bool_lazy_and_p:nn {
4975 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
4976 }{
4977 \tl_if_empty_p:V \l__hwexam_assign_due_tl
4978 }
4979 }{
4980 \bool_lazy_and_p:nn {
4981 \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl
4982 }{
4983 \tl_if_empty_p:V \l__hwexam_assign_due_tl
4984 }
4985 }{ ,~ }
4986
4987 \tl_if_empty:NTF \l__hwexam_inclasssign_due_tl {
4988 \tl_if_empty:NF \l__hwexam_assign_due_tl {
4989 \hwexam@due@kw\xspace \l__hwexam_assign_due_tl
4990 }
4991 }{
4992 \hwexam@due@kw\xspace \l__hwexam_inclasssign_due_tl
4993 }
4994
4995 \bool_lazy_all:nF {
4996 { \tl_if_empty_p:V \l__hwexam_inclasssign_given_tl }
4997 { \tl_if_empty_p:V \l__hwexam_assign_given_tl }
4998 { \tl_if_empty_p:V \l__hwexam_inclasssign_due_tl }
4999 { \tl_if_empty_p:V \l__hwexam_assign_due_tl }
5000 }{ #2 }
5001 }

```

`\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.

```

5002 \newcommand\assignment@title[3]{

```

```

5003 \tl_if_empty:NTF \l__hwexam_inclassassign_title_tl {
5004 \tl_if_empty:NTF \l__hwexam_assign_title_tl {
5005 #1
5006 }{
5007 #2\l__hwexam_assign_title_tl#3
5008 }
5009 }{
5010 #2\l__hwexam_inclassassign_title_tl#3
5011 }
5012 }

```

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

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

```

5013 \newcommand\assignment@number{
5014 \int_compare:nNnTF \l__hwexam_inclassassign_number_int = {-1} {
5015 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5016 \int_use:N \l__hwexam_assign_number_int
5017 }
5018 }{
5019 \int_use:N \l__hwexam_inclassassign_number_int
5020 }
5021 }

```

(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.

```

5022 \newenvironment{assignment}[1][ ]{
5023 \__hwexam_assignment_args:n { #1 }
5024 %\sref@target
5025 \let\__hwexamnum\l__hwexam_assign_number_int
5026 \int_compare:nNnF \l__hwexam_assign_number_int = {-1} {
5027 \stepcounter{assignment}
5028 }{
5029 \setcounter{assignment}{\int_use:N\__hwexamnum}
5030 }
5031 \setcounter{problem}{0}
5032 \def\current@section@level{\document@hwexamtype}
5033 %\sref@label@id{\document@hwexamtype \thesection}
5034 \begin{@assignment}
5035 }{
5036 \end{@assignment}
5037 }

```

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

```

5038 \def\__hwexasstitle{
5039 \protect\document@hwexamtype~\arabic{assignment}
5040 \assignment@title{}\;{} \; -- \given@due{}\}
5041 }

```

```

5042 \ifmultiple
5043 \newenvironment{@assignment}{
5044 \bool_if:NTF \l__hwexam_assign_loadmodules_bool {
5045 \begin{omgroup}[loadmodules]{\__hwexasstitle}
5046 }{
5047 \begin{omgroup}{\__hwexasstitle}
5048 }
5049 }{
5050 \end{omgroup}
5051 }

```

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

```

5052 \else
5053 \newenvironment{@assignment}{
5054 \begin{center}\bf
5055 \Large\@title\strut\
5056 \document@hwexamtype~\arabic{assignment}\assignment@title{\;}{:};{\;\}
5057 \large\given@due{--\;}{\;}{--}
5058 \end{center}
5059 }{}
5060 \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.

```

5061 \keys_define:nn { hwexam / inclassignment } {
5062 %id .str_set_x:N = \l__hwexam_assign_id_str,
5063 number .int_set:N = \l__hwexam_inclassign_number_int,
5064 title .tl_set:N = \l__hwexam_inclassign_title_tl,
5065 type .tl_set:N = \l__hwexam_inclassign_type_tl,
5066 given .tl_set:N = \l__hwexam_inclassign_given_tl,
5067 due .tl_set:N = \l__hwexam_inclassign_due_tl,
5068 mhrepos .str_set_x:N = \l__hwexam_inclassign_mhrepos_str
5069 }
5070 \cs_new_protected:Nn \__hwexam_inclassignment_args:n {
5071 \int_set:Nn \l__hwexam_inclassign_number_int {-1}
5072 \tl_clear:N \l__hwexam_inclassign_title_tl
5073 \tl_clear:N \l__hwexam_inclassign_type_tl
5074 \tl_clear:N \l__hwexam_inclassign_given_tl
5075 \tl_clear:N \l__hwexam_inclassign_due_tl
5076 \str_clear:N \l__hwexam_inclassign_mhrepos_str
5077 \keys_set:nn { hwexam / inclassignment }{ #1 }
5078 }
5079 \__hwexam_inclassignment_args:n {}
5080
5081 \newcommand\inputassignment[2][ ]{
5082 \__hwexam_inclassignment_args:n { #1 }
5083 \str_if_empty:NTF \l__hwexam_inclassign_mhrepos_str {
5084 \input{#2}
5085 }{
5086 \stex_in_repository:nn{\l__hwexam_inclassign_mhrepos_str}{

```

```

5087 \input{\mhp\path{\l__hwexam_inclasssign_mhrepos_str}{#2}}
5088 }
5089 }
5090 \__hwexam_inclassassignment_args:n {}
5091 }
5092 \newcommand\includeassignment[2][ ]{
5093 \newpage
5094 \inputassignment[#1]{#2}
5095 }

```

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

35.4 Typesetting Exams

\quizheading

```

5096 \ExplSyntaxOff
5097 \newcommand\quizheading[1]{%
5098 \def\@tas{#1}%
5099 \large\noindent NAME: \hspace{8cm} MAILBOX:\[2ex]%
5100 \ifx\@tas\empty\else%
5101 \noindent TA:~\@for\@I:=\@tas\do{\Large$\Box$}\@I\hspace*{1em}}\[2ex]%
5102 \fi%
5103 }
5104 \ExplSyntaxOn

```

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

\testheading

```

5105 \keys_define:nn { hwexam / testheading } {
5106 min .tl_set:N = \l__hwexam_testheading_min_tl,
5107 duration .tl_set:N = \l__hwexam_testheading_duration_tl,
5108 reqpts .tl_set:N = \l__hwexam_testheading_reqpts_tl
5109 }
5110 \cs_new_protected:Nn \__hwexam_testheading_args:n {
5111 \tl_clear:N \l__hwexam_testheading_min_tl
5112 \tl_clear:N \l__hwexam_testheading_duration_tl
5113 \tl_clear:N \l__hwexam_testheading_reqpts_tl
5114 \keys_set:nn { hwexam / testheading }{ #1 }
5115 }
5116 \newenvironment{testheading}[1][ ]{
5117 \__hwexam_testheading_args:n{ #1 }
5118 \noindent\large{Name:~\hfill
5119 Matriculation Number:\hspace*{2cm}\strut}\[1ex]
5120 \begin{center}
5121 \Large\textbf{\@title}\[1ex]
5122 \large\@date\[3ex]
5123 \end{center}
5124 \textbf{You~have~
5125 \tl_if_empty:NTF \l__hwexam_testheading_duration_tl {
5126 \l__hwexam_testheading_min_tl~minutes
5127 }{
5128 \l__hwexam_testheading_duration_tl
5129 }~

```

```

5130 (sharp)~for~the~test
5131 };\
5132 Write~the~solutions~to~the~sheet.
5133 \par\noindent
5134 \newcount\check@time\check@time=\l__hwexam_testheading_min_tl
5135 \advance\check@time by -\theassignment@totalmin
5136 The~estimated~time~for~solving~this~exam~is~
5137 {\theassignment@totalmin}-minutes,~
5138 leaving~you~{\the\check@time}-minutes~for~revising~
5139 your~exam.
5140
5141 \par\noindent
5142 \newcount\bonus@pts\bonus@pts=\theassignment@totalpts
5143 \advance\bonus@pts by -\l__hwexam_testheading_reqpts_tl
5144 You~can~reach~{\theassignment@totalpts}-points~if~you~
5145 solve~all~problems.~You~will~only~need~
5146 {\l__hwexam_testheading_reqpts_tl}-points~for~a~perfect~score,~
5147 i.e.~\ {\the\bonus@pts}-points~are~bonus~points.
5148 \vfill
5149 \begin{center}
5150 {
5151 \Large\em You~have~ample~time,~so~take~it~slow~
5152 and~avoid~rushing~to~mistakes!\}[2ex]
5153 Different~problems~test~different~skills~and~
5154 knowledge,~so~do~not~get~stuck~on~one~problem.
5155 }
5156 \vfill\par\resizebox{\textwidth}{!}{\correction@table}\}[3ex]
5157 \end{center}
5158 }{
5159 \newpage
5160 }

```

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

\testspace

```

5161 \newcommand\testspace[1]{\iftest\vspace*{#1}\fi}

```

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

\testnewpage

```

5162 \newcommand\testnewpage{\iftest\newpage\fi}

```

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

\testemptypage

```

5163 \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.

```

5164 \@=problems}
5165 \renewcommand\@problem[3]{
5166 \stepcounter{assignment@probs}
5167 \def\__problemspts{#2}

```

```

5168 \ifx\__problemspts\@empty\else
5169 \addtocounter{assignment@totalpts}{#2}
5170 \fi
5171 \def\__problemsmin{#3}\ifx\__problemsmin\@empty\else\addtocounter{assignment@totalmin}{#3}\fi
5172 \xdef\correction@probs{\correction@probs & #1}%
5173 \xdef\correction@pts{\correction@pts & #2}
5174 \xdef\correction@reached{\correction@reached & }
5175 }
5176 \@@=hwexam

```

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

\correction@table This macro generates the correction table

```

5177 \newcounter{assignment@probs}
5178 \newcounter{assignment@totalpts}
5179 \newcounter{assignment@totalmin}
5180 \def\correction@probs{\correction@probs@kw}%
5181 \def\correction@pts{\correction@pts@kw}%
5182 \def\correction@reached{\correction@reached@kw}%
5183 \def\after@correction@table{}%
5184 \stepcounter{assignment@probs}
5185 \newcommand\correction@table{
5186 \resizebox{\textwidth}{!}{%
5187 \begin{tabular}{|l|*{\theassignment@probs}{c|}|l|}\hline%
5188 &\multicolumn{\theassignment@probs}{c|}|%|
5189 {\footnotesize\correction@forgrading@kw} &\\ \hline
5190 \correction@probs & \correction@sum@kw & \correction@grade@kw\\ \hline
5191 \correction@pts & \theassignment@totalpts & \\ \hline
5192 \correction@reached & & \[.7cm]\hline
5193 \end{tabular}}
5194 \ifx\after@correction@table\@empty\else\strut\par\noindent\after@correction@table\fi}
5195 \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\wierglas{{\wierfont\char65}}
\newcommand\denker{{\denkerfont\char65}}
\newcommand\uhr{{\uhrfont\char65}}
\newcommand\warnschild{{\warnschildfont\char 65}}
\newcommand\hardA{\warnschild}
\newcommand\longA{\uhr}
\newcommand\thinkA{\denker}
\newcommand\discussA{\wierglas}

```