

The \TeX 3 Package *

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

2021-12-17

Abstract

TODO

*Version 3.0 (last revised 2021-12-17)

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
	Structures	30
10	sTeX-Statements	31
10.1	Macros and Environments	31
11	sTeX-Metatheory	32
11.1	Symbols	32

III	Extensions	33
12	Tikzinput	34
12.1	Macros and Environments	34
13	document-structure.sty: Semantic Markup for Open Mathematical Documents in L^AT_EX	35
13.1	Introduction	35
13.2	The User Interface	36
13.2.1	Package and Class Options	36
13.2.2	Document Structure	36
13.2.3	Ignoring Inputs	37
13.2.4	Structure Sharing	38
13.2.5	Global Variables	38
13.2.6	Colors	39
13.3	Limitations	39
14	Slides and Course Notes	40
14.1	Introduction	40
14.2	The User Interface	40
14.2.1	Package Options	40
14.2.2	Notes and Slides	41
14.2.3	Header and Footer Lines of the Slides	42
14.2.4	Frame Images	42
14.2.5	Colors and Highlighting	43
14.2.6	Front Matter, Titles, etc.	43
14.2.7	Excursions	43
14.2.8	Miscellaneous	43
14.3	Limitations	43
15	problem.sty: An Infrastructure for formatting Problems	44
15.1	Introduction	44
15.2	The User Interface	44
15.2.1	Package Options	44
15.2.2	Problems and Solutions	45
15.2.3	Multiple Choice Blocks	46
15.2.4	Including Problems	46
15.2.5	Reporting Metadata	46
15.3	Limitations	46
IV	Implementation	48

16	<code>sTeX</code>-Basics Implementation	49
16.1	The <code>sTeX</code> Document Class	49
16.2	Preliminaries	49
16.3	Messages and logging	50
16.4	Persistence	51
16.5	HTML Annotations	51
16.6	Languages	54
16.7	Activating/Deactivating Macros	55
17	<code>sTeX</code>-MathHub Implementation	56
17.1	Generic Path Handling	56
17.2	PWD and <code>kpsewhich</code>	58
17.3	File Hooks and Tracking	59
17.4	MathHub Repositories	60
18	<code>sTeX</code>-References Implementation	66
18.1	Document URIs and URLs	66
18.2	Setting Reference Targets	68
18.3	Using References	69
19	<code>sTeX</code>-Modules Implementation	70
19.1	The module environment	73
19.2	Invoking modules	78
20	<code>sTeX</code>-Module Inheritance Implementation	80
20.1	SMS Mode	80
20.2	Inheritance	84
21	<code>sTeX</code>-Symbols Implementation	89
21.1	Symbol Declarations	89
21.2	Notations	95
22	<code>sTeX</code>-Terms Implementation	103
22.1	Symbol Invocations	103
22.2	Terms	105
22.3	Notation Components	112
23	<code>sTeX</code>-Structural Features Implementation	114
23.1	The feature environment	114
23.2	Features	116
24	<code>sTeX</code>-Statements Implementation	121
24.1	Definitions	122
24.2	Assertions	123
24.3	Examples	125
25	<code>sTeX</code>-Others Implementation	126
26	<code>sTeX</code>-Metatheory Implementation	127
27	Tikzinput Implementation	130

28 document-structure.sty Implementation	132
28.1 The OMDoc Class	132
28.2 Class Options	132
28.3 Beefing up the document environment	133
28.4 Implementation: OMDoc Package	133
28.5 Package Options	133
28.6 Document Structure	135
28.7 Front and Backmatter	138
28.8 Global Variables	140
29 MiKoSlides – Implementation	141
29.1 Class and Package Options	141
29.2 Notes and Slides	143
29.3 Header and Footer Lines	146
29.4 Frame Images	148
29.5 Colors and Highlighting	149
29.6 Sectioning	150
29.7 Excursions	152
30 The Implementation	154
30.1 Package Options	154
30.2 Problems and Solutions	155
30.3 Multiple Choice Blocks	159
30.4 Including Problems	161
30.5 Reporting Metadata	161

Part I
Manual

Chapter 1

Stuff

1.1 Modules

`\sTeX`
`\stex`

Both print this \TeX logo.

1.1.1 Semantic Macros and Notations

Semantic macros invoke a formally declared symbol.

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

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

Example 1

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

ab

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

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

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

Example 2

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

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

.

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

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

Example 3

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

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

.

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

Example 4

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

Multiplying again by b yields...

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

Example 5

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

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

¹EdNOTE: TODO

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

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

Example 6

```
\symdef[ args=2,op={+}]{add}{#1 \comp+ #2}
The operator  $\textcolor{teal}{\$}\textcolor{teal}{\textit{add}}\textcolor{teal}{\$}$  adds two elements, as in  $\textcolor{teal}{\$}\textcolor{teal}{\textit{add}}\textcolor{teal}{ab}\textcolor{teal}{\$}$ .
```

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

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

Example 7

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

$\textcolor{teal}{\textit{Multiplication}}$ (denoted by \cdot) is defined by...

The macro `\comp` as used everywhere above is responsible for highlighting, linking, and tooltips, and should be wrapped around the notation (or text) components that should be treated accordingly. While it is attractive to just wrap a whole notation, this would also wrap around e.g. the arguments themselves, so instead, the user is tasked with marking the notation components themselves.

The precise behaviour of `\comp` is governed by the macro `\@comp`, which takes two arguments: The tex code of the text (unexpanded) to highlight, and the URI of the current symbol. `\@comp` can be safely redefined to customize the behaviour.

The starred variant `\symdecl*{foo}` does not introduce a semantic macro, but still declares a corresponding symbol. `foo` (like any other symbol, for that matter) can then be accessed via `\STEXsymbol{foo}` or (if `foo` was declared in a module `Foo`) via `\STEXModule{Foo}?{foo}`.

both `\STEXsymbol` and `\STEXModule` take any arbitrary ending segment of a full URI to determine which symbol or module is meant. e.g. `\STEXsymbol{Foo?foo}` is also valid, as are e.g. `\STEXModule{path?Foo}?{foo}` or `\STEXsymbol{path?Foo?foo}`

There’s also a convient shortcut `\symref{?foo}{some text}` for `\STEXsymbol{?foo}![some text]`

Other Argument Types

So far, we have stated the arity of a semantic macro directly. This works if we only have “normal” (or more precisely: *i*-type) arguments. To make use of other argument types, instead of providing the arity numerically, we can provide it as a sequence of characters

representing the argument types – e.g. instead of writing `args=2`, we can equivalently write `args=ii`, indicating that the macro takes two i-type arguments.

Besides i-type arguments, \TeX has two other types, which we will discuss now.

The first are *binding* (b-type) arguments, representing variables that are *bound* by the operator. This is the case for example in the above `\forevery`-macro: The first argument is not actually an argument that the `forevery` “function” is “applied” to; rather, the first argument is a new variable (e.g. x) that is *bound* in the subsequent argument. More accurately, the macro should therefore have been implemented thusly:

```
\symdef[args=bi]{forevery}{\forall #1.\; #2}
```

b-type arguments are indistinguishable from i-type arguments within \TeX , but are treated very differently in OMDOC and by MMT. More interesting *within* \TeX are a-type arguments, which represent (associative) arguments of flexible arity, which are provided as comma-separated lists. This allows e.g. better representing the `\mult`-macro above:

Example 8

```
\symdef[ args=a]{mult}{#1}{#1 \comp\cdot #2}
$\mult{a,b,c,{d^e},f}$
```

$$a \cdot b \cdot c \cdot d^e \cdot f$$

As the example above shows, notations get a little more complicated for associative arguments. For every a-type argument, the `\notation`-macro takes an additional argument that declares how individual entries in an a-type argument list are aggregated. The first notation argument then describes how the aggregated expression is combined into the full representation.

For a more interesting example, consider a flexary operator for ordered sequences in ordered set, that taking arguments $\{a, b, c\}$ and `\mathbb{R}` prints $a \leq b \leq c \in \mathbb{R}$. This operator takes two arguments (an a-type argument and an i-type argument), aggregates the individuals of the associative argument using `\leq`, and combines the result with `\in` and the second argument thusly:

Example 9

```
\symdef[ args=ai]{numseq}{#1 \comp\in #2}{#1 \comp\leq #2}
$numseq{a,b,c}{\mathbb{R}}$
```

$$a \leq b \leq c \in \mathbb{R}$$

Finally, B-type arguments combine the functionalities of a and b, i.e. they represent flexary binding operator arguments.

2 3

²EDNOTE: what about e.g. `\int \int \int f dx dy dz`?

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

Precedences

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

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

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

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

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

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

For example:

Example 10

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

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

1.1.2 Archives and Imports

Namespaces

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

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

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

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

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

Paths in Import-Statements

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

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

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

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

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

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

Part II

Documentation

Chapter 2

sTeX-Basics

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

debug ($\langle\log\text{-}prefix\rangle*$) 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 $\langle content \rangle$ with

`property="stex:⟨property⟩", resource="⟨resource⟩".`

`\stex_annotate_invisible:n` adds the attributes

`stex:visible="false", style="display:none".`

`\stex_annotate_invisible:nnn` combines the functionality of both.

<code>stex_annotate_env</code>	<code>\begin{stex_annotate_env}{⟨property⟩}{⟨resource⟩}</code> $\langle content \rangle$ <code>\end{stex_annotate_env}</code> behaves like <code>\stex_annotate:nnn {⟨property⟩} {⟨resource⟩} {⟨content⟩}</code> .
--------------------------------	---

<code>\c_stex_languages_prop</code>
<code>\c_stex_language_abbrevs_prop</code>

Map language abbreviations to their full babel names and vice versa. e.g. `\c_stex_languages_prop{en}` yields `english`, and `\c_stex_language_abbrevs_prop{english}` yields `en`.

<code>\stex_deactivate_macro:Nn</code>	<code>\stex_deactivate_macro:Nn⟨cs⟩{⟨environments⟩}</code>
<code>\stex_reactivate_macro:N</code>	

Makes the macro $\langle cs \rangle$ throw an error, indicating that it is only allowed in the context of $\langle environments \rangle$.

`\stex_reactivate_macro:N⟨cs⟩` reactivates it again, i.e. this happens ideally in the $\langle begin \rangle$ -code of the associated environments.

<code>\MSC</code>	<code>\MSC{⟨msc⟩}</code>
-------------------	--------------------------

Designates the *math subject classifier* of the current module / file.

Chapter 3

STEX-MathHub

Code related to managing and using MathHub repositories, files, paths and related hooks and methods.

3.1 Macros and Environments

<code>\stex_kpsewhich:n</code>	<code>\stex_kpsewhich:n</code> executes <code>kpsewhich</code> and stores the return in <code>\l_stex_kpsewhich_return_str</code> . This does not require shell escaping.
--------------------------------	---

3.1.1 Files, Paths, URIs

<code>\stex_path_from_string:Nn</code>	<code>\stex_path_from_string:Nn</code> $\langle path-variable \rangle$ $\{ \langle string \rangle \}$
<code>\stex_path_from_string:(NV cn cV)</code>	

turns the $\langle string \rangle$ into a path by splitting it at `/`-characters and stores the result in $\langle path-variable \rangle$. Also applies `\stex_path_canonicalize:N`.

<code>\stex_path_to_string:NN</code>	The inverse; turns a path into a string and stores it in the second argument variable, or
<code>\stex_path_to_string:N</code>	leaves it in the input stream.

<code>\stex_path_canonicalize:N</code>	Canonicalizes the path provided; in particular, resolves <code>.</code> and <code>..</code> path segments.
--	--

<code>\stex_path_if_absolute_p:N</code>	\star
<code>\stex_path_if_absolute:NTF</code>	\star

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

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

`\g_stex_currentfile_seq`

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

Test 1

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

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

3.1.2 MathHub Archives

`\mathhub`

`\c_stex_mathhub_seq`

`\c_stex_mathhub_str`

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

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

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

`\l_stex_current_repository_prop`

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

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

Test 2

```

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

```

```

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

```

Chapter 4

sTeX-References

Code related to links and cross-references

4.1 Macros and Environments

Chapter 5

sTeX-Modules

Code related to Modules

5.1 Macros and Environments

`\l_stex_current_module_prop`

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

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

Additionally, it stores:

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

`\l_stex_all_modules_seq`

Stores full URIs for all modules currently in scope.

```
\g_stex_module_files_prop
\g_stex_modules_in_file_seq
```

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

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

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

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

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

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

```
\stex_add_constant_to_current_module:n
```

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

```
\stex_add_import_to_current_module:n
```

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

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

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

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

```
\stex_modules_current_namespace:
```

Computes the current namespace

Test 3

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

```

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

```

.

5.1.1 The module-environment

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

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

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

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

Test 4

```

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

```

```

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

```

.

Test 5

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

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

\STEXModule \STEXModule {*⟨fragment⟩*}

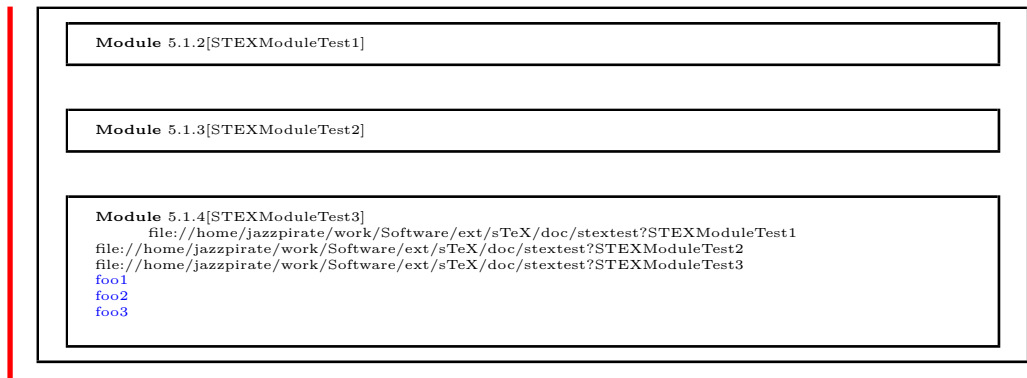
Attempts to find a module whose URI ends with *⟨fragment⟩* in the current scope and passes the full URI on to \stex_invoke_module:n.

\stex_invoke_module:n

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

Test 6

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



`\stex_activate_module:n`

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

Chapter 6

STEX-Module Inheritance

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

6.1 Macros and Environments

6.1.1 SMS Mode

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

`\g_stex_smsmode_allowedmacros_tl`

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

`\g_stex_smsmode_allowedmacros_escape_tl`

Macros that are executed with the category codes restored.

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

`\g_stex_smsmode_allowedenvs_seq`

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

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

Tests whether SMS mode is currently active.

`\stex_smsmode_set_codes:`

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

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

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

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

Test 7

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

6.1.2 Imports and Inheritance

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

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

Test 8

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

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

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

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

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

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

Test 14

```

\begin{module}{MathTest1}
\importmodule{Foo}
\notation[foo, prec=500;20x20x20]{bar}{\comp\langle {#1} ^ {#2} _{#3} \comp\rangle }
 $\bar{abc}$  and  $\bar{foo} abc$ .
\end{module}

```

Module 8.1.1[MathTest1]
 $\langle x20x20a^b{}_c \rangle$ 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} _{com
 $\bar{foo} a\{b,c,d,e,f\}g$  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

TEX-Structural Features

Code related to structural features

9.1 Macros and Environments

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]
 $a\circ b:M$
file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1/Magma-feature?universe,file://Home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma?universe
feature?op
»macro:->\stex_invoke_symbol:n {file://home/jazzpirate/work/Software/ext/sTeX/doc/stextest?StructureTest1?Magma}<
Test: $a+b$
Test2: $\langle U,+ \rangle$

Chapter 10

TeX-Statements

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

10.1 Macros and Environments

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

Chapter 11

sTeX-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.

11.1 Symbols

Part III
Extensions

Chapter 12

Tikzinput

12.1 Macros and Environments

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

Chapter 13

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

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

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

13.1 Introduction

$\S\mathrm{T}_\mathrm{E}\mathrm{X}$ is a version of $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ that allows to markup $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ documents semantically without leaving the document format, essentially turning $\mathrm{T}_\mathrm{E}\mathrm{X}/\mathrm{L}^{\mathrm{A}}\mathrm{T}_\mathrm{E}\mathrm{X}$ into a document format for mathematical knowledge management (MKM). The package supports direct translation to the OMDOC format [Koh06]

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

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

source and the formatter does the copying during document formatting/presentation.⁴

13.2 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`.

13.2.1 Package and Class Options

The `omdoc` class accept the following options:

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

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

13.2.2 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 \LaTeX ML transformation.

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

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

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

⁴EDNOTE: 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 1 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 1: 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.

13.2.3 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].

13.2.4 Structure Sharing

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

The `\STRlabel` macro has a variant `\STRsemantics`, where the label argument is optional, and which takes a third argument, which is ignored in \LaTeX . This allows to specify the meaning of the content (whatever that may mean) in cases, where the source document is not formatted for presentation, but is transformed into some content markup format.⁵

13.2.5 Global Variables

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

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

⁵EdNOTE: document LMID und LMXRef here if we decide to keep them.

13.2.6 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.

13.3 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 14

Slides and Course Notes

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

14.1 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`.

14.2 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.

14.2.1 Package Options

The `mikoslides` class takes a variety of class options:⁶

- | | |
|---------------------------|---|
| <code>slides</code> | • The options <code>slides</code> and <code>notes</code> switch between slides mode and notes mode (see |
| <code>notes</code> | Section 14.2.2). |
| <code>sectocframes</code> | • 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 14.2.4). 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>.

14.2.2 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 2: A typical Course Notes File

By interleaving the `frame` and `note` environments, we can build course notes as shown in Figure 2.

`\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.

⁶EDNOTE: 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.

14.2.3 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.

14.2.4 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.⁷

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

⁷EdNOTE: 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}`

14.2.5 Colors and Highlighting

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

14.2.6 Front Matter, Titles, etc.

14.2.7 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}
```

14.2.8 Miscellaneous

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 [sTeX 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. This is a problem of the underlying `omdoc` package.

Chapter 15

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.

15.1 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.

15.2 The User Interface

15.2.1 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

15.2.2 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 3 and the resulting markup see Figure 4.

```
\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 3: 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()
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 4: The Formatted Problem from Figure 3

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.

15.2.3 Multiple Choice Blocks

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

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

See Figure ?? for an example

15.2.4 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.

15.2.5 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.

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. 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()

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

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

Problem0()

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 5: A Problem with a multiple choice block

Part IV

Implementation

Chapter 16

ST_EX -Basics Implementation

16.1 The ST_EXDocument Class

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

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

16.2 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 }
31 \ProcessKeysOptions { stex }

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

```

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

16.3 Messages and logging

```

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

\stex_debug:nn A simple macro issuing package messages with subpath.
50 \cs_new_protected:Nn \stex_debug:nn {
51   \clist_if_in:NnTF \c_stex_debug_clist { all } {
52     \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
53       \Debug~#1:~#2\\
54     }
55     \msg_none:nn{stex}{debug / #1}
56   }{
57     \clist_if_in:NnT \c_stex_debug_clist { #1 } {
58       \exp_args:Nnnx\msg_set:nnn{stex}{debug / #1}{
59         \Debug~#1:~#2\\
60       }
61       \msg_none:nn{stex}{debug / #1}
62     }
63   }
64 }

```

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

Redirecting messages:

```

65 \clist_if_in:NnTF \c_stex_debug_clist {all} {

```

```

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

```

16.4 Persistence

```

74 <@@=stex_persist>

```

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

```

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

```

(End definition for \c_stex_persist_sms_iow.)

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

```

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

```

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

16.5 HTML Annotations

```

93 <@@=stex_annotate>
94 \RequirePackage{scalatex}

```

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

```

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

```

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

```

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

```



```

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

```

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

```

(End definition for \l__stex_annotate_arg_tl and \c__stex_annotate_emptyarg_tl.)

```

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

```

(End definition for __stex_annotate_checkempty:n.)

\l_stex_html_do_output_bool Whether to (locally) produce HTML output

```

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

```

(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

```

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

```

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

`\stex_annotate:env`

`\stex_annotate_invisible:n`

`\stex_annotate_invisible:nnn`

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

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

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

```

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

```

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

16.6 Languages

```

218 <@@=stex_language>

```

`\c_stex_languages_prop`
`\c_stex_language_abbrevs_prop`

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

```

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

```

```

230
231 \prop_const_from_keyval:Nn \c_stex_language_abbrevs_prop {
232   english   = en ,
233   ngerman   = de ,
234   arabic    = ar ,
235   bulgarian = bg ,
236   russian   = ru ,
237   finnish   = fi ,
238   romanian  = ro ,
239   turkish   = tr ,
240   french    = fr
241 }
242 % todo: chinese simplified (zhs)
243 %       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:

```

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

```

16.7 Activating/Deactivating Macros

`\stex_deactivate_macro:Nn`

```

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

```

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

`\stex_reactivate_macro:N`

```

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

```

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

```

265 \</package>

```

Chapter 17

STEX -MathHub Implementation

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

17.1 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
281 \cs_new_protected:Nn \stex_path_from_string:Nn {
282   \str_set:Nx \l_tmpa_str { #2 }
283   \str_if_empty:NTF \l_tmpa_str {
284     \seq_clear:N #1
285   }{
286     \exp_args:NNNo \seq_set_split:Nnn #1 / { \l_tmpa_str }
287     \sys_if_platform_windows:T{
288       \seq_clear:N \l_tmpa_tl
289       \seq_map_inline:Nn #1 {
290         \seq_set_split:Nnn \l_tmpb_tl \c_backslash_str { ##1 }
291         \seq_concat:NNN \l_tmpa_tl \l_tmpa_tl \l_tmpb_tl

```

```

292     }
293     \seq_set_eq:NN #1 \l_tmpa_tl
294   }
295   \stex_path_canonicalize:N #1
296 }
297 }
298 \cs_generate_variant:Nn \stex_path_from_string:Nn
299 { 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
300 \cs_new_protected:Nn \stex_path_to_string:NN {
301   \exp_args:NNe \str_set:Nn #2 { \seq_use:Nn #1 / }
302 }
303
304 \cs_new:Nn \stex_path_to_string:N {
305   \seq_use:Nn #1 /
306 }

```

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

```

\c__stex_path_dot_str . and .., respectively.
\c__stex_path_up_str
307 \str_const:Nn \c__stex_path_dot_str {.}
308 \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.

```

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

```

```

333     }
334   }{
335     \str_if_empty:NF \l_tmpa_tl {
336       \exp_args:NNo \seq_put_right:Nn \l_tmpa_seq { \l_tmpa_tl }
337     }
338   }
339 }
340 }
341 \seq_gset_eq:NN #1 \l_tmpa_seq
342 }
343 }

```

(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`

```

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

```

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

17.2 PWD and kpsewhich

`\stex_kpsewhich:n`

```

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

```

(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`

```

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

17.3 File Hooks and Tracking

371 `<@@=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

372 `\seq_gclear_new:N\g__stex_files_stack`

(End definition for `\g__stex_files_stack`.)

`\c_stex_mainfile_seq`

`\c_stex_mainfile_str`

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

374 `\stex_path_from_string:Nn \c_stex_mainfile_seq`

375 `\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`.

```

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

```

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

17.4 MathHub Repositories

```

400 <@=stex_mathhub>

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

```

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

```

(End definition for `__stex_mathhub_do_manifest:n`.)

```

\l__stex_mathhub_manifest_file_seq
437 \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`:

```

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

```

(End definition for `_stex_mathhub_find_manifest:N`.)

`\c_stex_mathhub_manifest_ior` File variable used for MANIFEST-files

```

473 \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:

```

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

```

```

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

```

(End definition for `__stex_mathhub_parse_manifest:n.`)

`\stex_set_current_repository:n`

```

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

```

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

`\stex_require_repository:n`

```

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

```

```

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

```

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

`\l_stex_current_repository_prop` Current MathHub repository

```

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

```

(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.

```

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

```

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

\inputref
\inputref:nn

```

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

```

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

\mhp

```

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

```

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

\libinput

```

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

```

```

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

```

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

```

630 </package>

```

Chapter 18

STEX -References Implementation

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

18.1 Document URIs and URLs

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

```

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

```



```

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

```

18.2 Setting Reference Targets

```

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

```

```

766   }
767 }
768 \cs_new_protected:Nn \stex_ref_new_sym_target:n {
769   \str_gset_eq:cN {sref_sym_#1_uri} \l_stex_current_docns_str
770 }

```

18.3 Using References

```

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

```

Chapter 19

STEX -Modules Implementation

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

`\l_stex_current_module_prop` The current module:

```
804 \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

```
805 \seq_new:N \l_stex_all_modules_seq
```

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

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

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

```

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

```

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

```

\stex_add_to_current_module:n
\STEXexport
816 \cs_new_protected:Nn \stex_add_to_current_module:n {
817   \prop_get:NnN \l_stex_current_module_prop { content } \l_tmpa_tl
818   \tl_put_right:Nn \l_tmpa_tl { #1 }
819   \prop_put:Nno \l_stex_current_module_prop { content } { \l_tmpa_tl }
820 }
821 \cs_new_protected:Npn \STEXexport #1 {
822   #1
823   \stex_add_to_current_module:n { #1 }
824   \stex_smsmode_set_codes:
825 }
826 \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
827 \cs_new_protected:Nn \stex_add_constant_to_current_module:n {
828   \str_set:Nx \l_tmpa_str { #1 }
829   \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
830   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
831   \prop_put:Nno \l_stex_current_module_prop { constants } \l_tmpa_seq
832 }

```

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

```

\stex_add_import_to_current_module:n
833 \cs_new_protected:Nn \stex_add_import_to_current_module:n {
834   \str_set:Nx \l_tmpa_str { #1 }
835   \prop_get:NnN \l_stex_current_module_prop { imports } \l_tmpa_seq
836   \seq_put_right:No \l_tmpa_seq { \l_tmpa_str }
837   \prop_put:Nno \l_stex_current_module_prop { imports } \l_tmpa_seq
838 }

```

(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).

```

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

```

(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`

```

868 \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.

```

869 \cs_new_protected:Nn \stex_modules_current_namespace: {
870   \prop_get:NnNTF \l_stex_current_repository_prop { ns } \l_tmpa_str {
871     \stex_modules_compute_namespace:nN \l_tmpa_str \g_stex_currentfile_seq
872   }{
873     % split off file extension
874     \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
875     \seq_pop_right:NN \l_tmpa_seq \l_tmpb_str
876     \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq . \l_tmpb_str
877     \seq_get_left:NN \l_tmpb_seq \l_tmpb_str
878     \seq_put_right:No \l_tmpa_seq \l_tmpb_str
879     \str_set:Nx \l_stex_modules_ns_str {
880       file:/\stex_path_to_string:N \l_tmpa_seq

```

```

881     }
882   }
883 }

```

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

19.1 The module environment

module arguments:

```

884 \keys_define:nn { stex / module } {
885   title      .str_set_x:N = \l_stex_module_title_str ,
886   ns         .str_set_x:N = \l_stex_module_ns_str ,
887   lang       .str_set_x:N = \l_stex_module_lang_str ,
888   sig        .str_set_x:N = \l_stex_module_sig_str ,
889   creators   .str_set_x:N = \l_stex_module_creators_str ,
890   contributors .str_set_x:N = \l_stex_module_contributors_str ,
891   meta       .str_set_x:N = \l_stex_module_meta_str
892 }
893
894 \cs_new_protected:Nn \__stex_modules_args:n {
895   \str_clear:N \l_stex_module_title_str
896   \str_clear:N \l_stex_module_ns_str
897   \str_clear:N \l_stex_module_lang_str
898   \str_clear:N \l_stex_module_sig_str
899   \str_clear:N \l_stex_module_creators_str
900   \str_clear:N \l_stex_module_contributors_str
901   \str_clear:N \l_stex_module_meta_str
902   \keys_set:nn { stex / module } { #1 }
903 }
904
905 % module parameters here? In the body?
906

```

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

```

907 \cs_new_protected:Nn \stex_module_setup:nn {
908   \str_set:Nx \l_stex_module_name_str { #2 }
909   \__stex_modules_args:n { #1 }
910
911   First, we set up the name and namespace of the module.
912   Are we in a nested module?
913
914   \stex_if_in_module:TF {
915     % Nested module
916     \prop_get:NnN \l_stex_current_module_prop
917       { ns } \l_stex_module_ns_str
918     \str_set:Nx \l_stex_module_name_str {
919       \prop_item:Nn \l_stex_current_module_prop
920       { name } / \l_stex_module_name_str
921     }
922   }{
923     % not nested:
924     \str_if_empty:NT \l_stex_module_ns_str {
925       \stex_modules_current_namespace:
926       \str_set_eq:NN \l_stex_module_ns_str \l_stex_modules_ns_str
927     }
928   }
929 }

```

```

923 \exp_args:NNNo \seq_set_split:Nnn \l_tmpa_seq
924 / {\l_stex_module_ns_str}
925 \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
926 \str_if_eq:NNT \l_tmpa_str \l_stex_module_name_str {
927   \str_set:Nx \l_stex_module_ns_str {
928     \stex_path_to_string:N \l_tmpa_seq
929   }
930 }
931 }
932 }

```

Next, we determine the language of the module:

```

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

```

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

```

955 \str_if_empty:NTF \l_stex_module_sig_str {
956   \str_clear:N \l_tmpa_str
957   \seq_clear:N \l_tmpa_seq
958   \tl_clear:N \l_tmpa_tl
959   \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
960     name      = \l_stex_module_name_str ,
961     ns        = \l_stex_module_ns_str ,
962     imports   = \exp_not:o { \l_tmpa_seq } ,
963     constants = \exp_not:o { \l_tmpa_seq } ,
964     content   = \exp_not:o { \l_tmpa_tl } ,
965     file      = \exp_not:o { \g_stex_currentfile_seq } ,
966     lang      = \l_stex_module_lang_str ,
967     sig       = \l_stex_module_sig_str ,
968     meta      = \l_stex_module_meta_str
969   }
970 }{
971   \str_if_empty:NT \l_stex_module_lang_str {

```

```

972     \msg_error:nnnn{stex}{error/siglanguage}{
973       \l_stex_module_ns_str?\l_stex_module_name_str
974     }{\l_stex_module_sig_str}
975   }
976
977   \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
978   \seq_pop_right:NN \l_tmpa_seq \l_tmpa_str
979   \seq_set_split:NnV \l_tmpb_seq . \l_tmpa_str
980   \seq_pop_right:NN \l_tmpb_seq \l_tmpa_str % .tex
981   \seq_pop_left:NN \l_tmpb_seq \l_tmpa_str % <filename>
982   \str_set:Nx \l_tmpa_str {
983     \stex_path_to_string:N \l_tmpa_seq /
984     \l_tmpa_str . \l_stex_module_sig_str .tex
985   }
986   \IfFileExists \l_tmpa_str {
987     \exp_args:No \stex_in_smsmode:nn { \l_tmpa_str } {
988       \seq_clear:N \l_stex_all_modules_seq
989       \prop_clear:N \l_stex_current_module_prop
990       \stex_debug:nn{modules}{Loading~signature~\l_tmpa_str}
991       \input { \l_tmpa_str }
992     }
993   }{
994     \msg_error:nnn{stex}{error/unknownmodule}{for~signature~\l_tmpa_str}
995   }
996   \stex_activate_module:n {
997     \l_stex_module_ns_str ? \l_stex_module_name_str
998   }
999   \prop_set_eq:Nc \l_stex_current_module_prop {
1000     c_stex_module_
1001     \l_stex_module_ns_str ?
1002     \l_stex_module_name_str
1003     _prop
1004   }
1005 }

```

We load the metatheory:

```

1006 \str_if_empty:NT \l_stex_module_meta_str {
1007   \str_set:Nx \l_stex_module_meta_str {
1008     \c_stex_metatheory_ns_str ? Metatheory
1009   }
1010 }
1011 \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1012   \exp_args:Nx \stex_add_to_current_module:n {
1013     \stex_activate_module:n {\l_stex_module_meta_str}
1014   }
1015   \stex_activate_module:n {\l_stex_module_meta_str}
1016 }
1017 }

```

(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}`


```

1018 \cs_new_protected:Nn \__stex_modules_begin_module:nn {
1019   \stex_reactivate_macro:N \STEXexport
1020   \stex_reactivate_macro:N \importmodule
1021   \stex_reactivate_macro:N \symdecl
1022   \stex_reactivate_macro:N \notation
1023   \stex_reactivate_macro:N \symdef
1024   \stex_module_setup:nn{#1}{#2}
1025
1026   \stex_debug:nn{modules}{
1027     New~module:\\
1028     Namespace:~\l_stex_module_ns_str\\
1029     Name:~\l_stex_module_name_str\\
1030     Language:~\l_stex_module_lang_str\\
1031     Signature:~\l_stex_module_sig_str\\
1032     Metatheory:~\l_stex_module_meta_str\\
1033     File:~\stex_path_to_string:N \g_stex_currentfile_seq
1034   }
1035
1036   \seq_put_right:Nx \l_stex_all_modules_seq {
1037     \l_stex_module_ns_str ? \l_stex_module_name_str
1038   }
1039
1040   \seq_gput_right:Nx \g_stex_modules_in_file_seq
1041     { \l_stex_module_ns_str ? \l_stex_module_name_str }
1042
1043   \stex_if_smsmode:TF {
1044     \stex_smsmode_set_codes:
1045   } {
1046     \begin{stex_annotate_env} {theory} {
1047       \l_stex_module_ns_str ? \l_stex_module_name_str
1048     }
1049
1050     \stex_annotate_invisible:nnn{header}{} {
1051       \stex_annotate:nnn{language}{ \l_stex_module_lang_str }{}
1052       \stex_annotate:nnn{signature}{ \l_stex_module_sig_str }{}
1053       \str_if_eq:VnF \l_stex_module_meta_str {NONE} {
1054         \stex_annotate:nnn{metatheory}{ \l_stex_module_meta_str }{}
1055       }
1056     }
1057   }
1058   % TODO: Inherit metatheory for nested modules?
1059 }
1060 \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}

```

1061 \cs_new_protected:Nn \__stex_modules_end_module: {
1062   \str_set:Nx \l_tmpa_str {
1063     c_stex_module_
1064     \prop_item:Nn \l_stex_current_module_prop { ns } ?
1065     \prop_item:Nn \l_stex_current_module_prop { name }
1066     _prop
1067   }

```

```

1068 %^^A \prop_new:c { \l_tmpa_str }
1069 \prop_gset_eq:cn { \l_tmpa_str } \l_stex_current_module_prop
1070 \stex_debug:nn{modules}{Closing module~\prop_item:Nn \l_stex_current_module_prop { name }}
1071 }

```

(End definition for `_stex_modules_end_module:.`)

@module The core environment, with no header

```

1072 \iffalse \begin{stex_annotate_env} \fi %^^A make syntax highlighting work again
1073 \NewDocumentEnvironment { @module } { 0{} m } {
1074   \par
1075   \_stex_modules_begin_module:nn{#1}{#2}
1076 } {
1077   \_stex_modules_end_module:
1078   \stex_if_smsmode:TF {
1079     \exp_args:Nx \stex_add_to_sms:n {
1080       \prop_gset_from_keyval:cn {
1081         c_stex_module_
1082         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1083         \prop_item:Nn \l_stex_current_module_prop { name }
1084         _prop
1085       } {
1086         name      = \prop_item:cn { \l_tmpa_str } { name } ,
1087         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
1088         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
1089         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
1090         content   = \prop_item:cn { \l_tmpa_str } { content } ,
1091         file      = \prop_item:cn { \l_tmpa_str } { file } ,
1092         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
1093         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
1094         meta      = \prop_item:cn { \l_tmpa_str } { meta }
1095       }
1096     }
1097   }{
1098     \end{stex_annotate_env}
1099   }
1100 }

```

\stex_modules_heading: Code for document headers

```

1101 \cs_if_exist:NTF \thesection {
1102   \newcounter{module}[section]
1103 }{
1104   \newcounter{module}
1105 }
1106
1107 \bool_if:NT \c_stex_showmods_bool {
1108   \latexml_if:F { \RequirePackage{mdframed} }
1109 }
1110
1111 \cs_new_protected:Nn \stex_modules_heading: {
1112   \stepcounter{module}
1113   \par
1114   \bool_if:NT \c_stex_showmods_bool {
1115     \noindent{\textbf{Module}} ~

```

```

1116     \cs_if_exist:NT \thesection {\thesection.}
1117     \themodule ~ [\l_stex_module_name_str]
1118   }
1119   \str_if_empty:NTF \l_stex_module_title_str {
1120   }{
1121     \quad(\l_stex_module_title_str)\hfill
1122   }\par
1123 }
1124 \edef\@currentlabel{Module~\thesection.\themodule~[\l_stex_module_name_str]}
1125 % TODO
1126 \stex_ref_new_doc_target:n \l_stex_module_name_str
1127 }

```

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

Finally:

```

1128 \NewDocumentEnvironment { module } { 0{} m } {
1129   \bool_if:NT \c_stex_showmods_bool {
1130     \begin{mdframed}
1131   }
1132   \begin{@module} [#1] {#2}
1133   \stex_modules_heading:
1134 }{
1135   \end{@module}
1136   \bool_if:NT \c_stex_showmods_bool {
1137     \end{mdframed}
1138   }
1139 }

```

19.2 Invoking modules

```

\STEXModule
\stex_invoke_module:n
1140 \NewDocumentCommand \STEXModule { m } {
1141   \exp_args:NNx \str_set:Nn \l_tmpa_str { #1 }
1142   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
1143   \tl_set:Nn \l_tmpa_tl {
1144     \msg_error:nnn{stex}{error/unknownmodule}{#1}
1145   }
1146   \seq_map_inline:Nn \l_stex_all_modules_seq {
1147     \str_set:Nn \l_tmpb_str { ##1 }
1148     \str_if_eq:eeT { \l_tmpa_str } {
1149       \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1150     } {
1151       \seq_map_break:n {
1152         \tl_set:Nn \l_tmpa_tl {
1153           \stex_invoke_module:n { ##1 }
1154         }
1155       }
1156     }
1157   }
1158   \l_tmpa_tl
1159 }
1160
1161 \cs_new_protected:Nn \stex_invoke_module:n {

```

```

1162 \stex_debug:nn{modules}{Invoking~module~#1}
1163 \peek_charcode_remove:NTF ! {
1164   \__stex_modules_invoke_uri:nN { #1 }
1165 } {
1166   \peek_charcode_remove:NTF ? {
1167     \__stex_modules_invoke_symbol:nn { #1 }
1168   } {
1169     \msg_error:nnn{stex}{error/syntax}{
1170       ?~or~!~expected~after~
1171       \c_backslash_str STEXModule{#1}
1172     }
1173   }
1174 }
1175 }
1176
1177 \cs_new_protected:Nn \__stex_modules_invoke_uri:nN {
1178   \str_set:Nn #2 { #1 }
1179 }
1180
1181 \cs_new_protected:Nn \__stex_modules_invoke_symbol:nn {
1182   \stex_invoke_symbol:n{#1?#2}
1183 }

```

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

\stex_activate_module:n

```

1184 \cs_new_protected:Nn \stex_activate_module:n {
1185   \stex_debug:nn{modules}{Activating~module~#1}
1186   \exp_args:NNx \seq_if_in:NnF \l_stex_all_modules_seq { #1 } {
1187     \seq_put_right:Nx \l_stex_all_modules_seq { #1 }
1188     \prop_item:cn { c_stex_module_#1_prop } { content }
1189   }
1190 }

```

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

```

1191 </package>

```

Chapter 20

STEX -Module Inheritance Implementation

```
1192 <*package>
1193
1194 %%%%%%%%%% inheritance.dtx %%%%%%%%%%
1195
```

20.1 SMS Mode

```
1196 <@@=stex_smsmode>

\g_stex_smsmode_allowedmacros_tl
\g_stex_smsmode_allowedmacros_escape_tl
\g_stex_smsmode_allowedenvs_seq

1197 \tl_new:N \g_stex_smsmode_allowedmacros_tl
1198 \tl_new:N \g_stex_smsmode_allowedmacros_escape_tl
1199 \seq_new:N \g_stex_smsmode_allowedenvs_seq
1200
1201 \tl_set:Nn \g_stex_smsmode_allowedmacros_tl {
1202   \makeatletter
1203   \makeatother
1204   \ExplSyntaxOn
1205   \ExplSyntaxOff
1206 }
1207
1208 \tl_set:Nn \g_stex_smsmode_allowedmacros_escape_tl {
1209   \symdef
1210   \importmodule
1211   \notation
1212   \symdecl
1213   \STEXexport
1214 }
1215
1216 \exp_args:NNx \seq_set_from_clist:Nn \g_stex_smsmode_allowedenvs_seq {
1217   \tl_to_str:n {
1218     module,
1219     @module
```

```

1220 }
1221 }

```

(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

```

```

1222 \bool_new:N \g__stex_smsmode_bool
1223 \bool_set_false:N \g__stex_smsmode_bool
1224 \prg_new_conditional:Nnn \stex_if_smsmode: { p, T, F, TF } {
1225   \bool_if:NTF \g__stex_smsmode_bool \prg_return_true: \prg_return_false:
1226 }

```

(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

```

```

1227 \bool_new:N \g__stex_smsmode_catcode_bool
1228 \bool_set_false:N \g__stex_smsmode_catcode_bool
1229 \prg_new_conditional:Nnn \__stex_smsmode_if_catcodes: { p, T, F, TF } {
1230   \bool_if:NTF \g__stex_smsmode_catcode_bool
1231   \prg_return_true: \prg_return_false:
1232 }

```

(End definition for `__stex_smsmode_if_catcodes:TF`.)

```

\stex_smsmode_set_codes:

```

```

1233 \cs_new_protected:Nn \stex_smsmode_set_codes: {
1234   \stex_if_smsmode:T {
1235     \__stex_smsmode_if_catcodes:F {
1236       \bool_gset_true:N \g__stex_smsmode_catcode_bool
1237       \exp_after:wN \char_gset_active_eq:NN
1238       \c_backslash_str \__stex_smsmode_cs:
1239       \tex_global:D \char_set_catcode_active:N \
1240       \tex_global:D \char_set_catcode_other:N $
1241       \tex_global:D \char_set_catcode_other:N ^
1242       \tex_global:D \char_set_catcode_other:N _
1243       \tex_global:D \char_set_catcode_other:N &
1244       \tex_global:D \char_set_catcode_other:N ##
1245     }
1246   }
1247 } \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.

```

1248 \cs_new_protected:Nn \__stex_smsmode_unset_codes: {
1249   \__stex_smsmode_if_catcodes:T {
1250     \bool_gset_false:N \g__stex_smsmode_catcode_bool
1251     \exp_after:wN \tex_global:D \exp_after:wN
1252     \char_set_catcode_escape:N \c_backslash_str
1253     \tex_global:D \char_set_catcode_math_toggle:N $
1254     \tex_global:D \char_set_catcode_math_superscript:N ^
1255     \tex_global:D \char_set_catcode_math_subscript:N _
1256     \tex_global:D \char_set_catcode_alignment:N &
1257     \tex_global:D \char_set_catcode_parameter:N ##
1258   }
1259 } \iffalse $ \fi % to make syntax highlighting work again

```

(End definition for `_stex_smsmode_unset_codes:`.)

`\stex_in_smsmode:nn`

```

1260 \cs_new_protected:Nn \stex_in_smsmode:nn {
1261   \vbox_set:Nn \l_tmpa_box {
1262     \bool_set_eq:cN { l__stex_smsmode_#1_bool } \g__stex_smsmode_bool
1263     \bool_gset_true:N \g__stex_smsmode_bool
1264     \stex_smsmode_set_codes:
1265     #2
1266     \bool_gset_eq:Nc \g__stex_smsmode_bool { l__stex_smsmode_#1_bool }
1267     \stex_if_smsmode:F {
1268       \__stex_smsmode_unset_codes:
1269     }
1270   }
1271   \box_clear:N \l_tmpa_box
1272 }

```

(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:

```

1273 \cs_new_protected:Nn \_stex_smsmode_cs: {
1274   \str_clear:N \l_tmpa_str
1275   \peek_analysis_map_inline:n {
1276     % #1: token (one expansion)
1277     % #2: charcode
1278     % #3 catcode
1279     \token_if_eq_charcode:NNTF ##3 B {
1280       % token is a letter
1281       \exp_args:NNNo \str_put_right:Nn \l_tmpa_str { ##1 }
1282     } {
1283       \str_if_empty:NTF \l_tmpa_str {
1284         % we don't allow (or need) single non-letter CSs
1285         % for now
1286         \peek_analysis_map_break:
1287       }{
1288         \str_if_eq:onTF \l_tmpa_str { begin } {
1289           \peek_analysis_map_break:n {
1290             \exp_after:wN \_stex_smsmode_checkbegin:n ##1
1291           }
1292         } {
1293           \str_if_eq:onTF \l_tmpa_str { end } {
1294             \peek_analysis_map_break:n {
1295               \exp_after:wN \_stex_smsmode_checkend:n ##1
1296             }
1297           } {
1298             \tl_set:Nn \l_tmpa_tl { \use:c{\l_tmpa_str} }
1299             \exp_args:NNNo \exp_args:NNNo \tl_if_in:NnTF
1300             \g_stex_smsmode_allowedmacros_tl
1301             { \use:c{\l_tmpa_str} } {
1302               \stex_debug:nn{modules}{Executing-1:~\l_tmpa_str}
1303               \peek_analysis_map_break:n {
1304                 \exp_after:wN \l_tmpa_tl ##1
1305               }

```

```

1306     } {
1307         \exp_args:NNo \exp_args:NNo \tl_if_in:NnTF
1308         \g_stex_smsmode_allowedmacros_escape_tl
1309         { \use:c{\l_tmpa_str} } {
1310             \__stex_smsmode_unset_codes:
1311             \stex_debug:nn{modules}{Executing~2:~\l_tmpa_str}
1312             % TODO \__stex_smsmode_rescan_cs:
1313             \int_compare:nNnTF {##2} = {92} {
1314                 \peek_analysis_map_break:n {
1315                     \__stex_smsmode_unset_codes:
1316                     \__stex_smsmode_rescan_cs:
1317                 }
1318             } {
1319                 \peek_analysis_map_break:n {
1320                     \exp_after:wN \l_tmpa_tl ##1
1321                 }
1322             }
1323         } {
1324             \int_compare:nNnTF {##2} = {92} {
1325                 \peek_analysis_map_break:n { \__stex_smsmode_cs: }
1326             } {
1327                 \peek_analysis_map_break:n { \exp_after:wN\relax ##1 }
1328             }
1329         }
1330     }
1331 }
1332 }
1333 }
1334 }
1335 }
1336 }

```

(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:

```

1337 \cs_new_protected:Nn \__stex_smsmode_rescan_cs: {
1338     \str_clear:N \l_tmpb_str
1339     \peek_analysis_map_inline:n {
1340         \token_if_eq_charcode:NNTF ##3 B {
1341             % token is a letter
1342             \exp_args:NNo \str_put_right:Nn \l_tmpb_str { ##1 }
1343         } {
1344             \peek_analysis_map_break:n {
1345                 \exp_after:wN \use:c \exp_after:wN {
1346                     \exp_after:wN \l_tmpa_str\exp_after:wN
1347                 } \use:c { \l_tmpb_str \exp_after:wN } ##1
1348             }
1349         }
1350     }
1351 }

```

(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.

```

1352 \cs_new_protected:Nn \__stex_smsmode_checkbegin:n {
1353   \str_set:Nn \l_tmpa_str { #1 }
1354   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1355     \__stex_smsmode_unset_codes:
1356     \begin{#1}
1357   }
1358 }
```

(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.

```

1359 \cs_new_protected:Nn \__stex_smsmode_checkend:n {
1360   \str_set:Nn \l_tmpa_str { #1 }
1361   \seq_if_in:NoT \g_stex_smsmode_allowedenvs_seq \l_tmpa_str {
1362     \end{#1}
1363   }
1364 }
```

(End definition for `__stex_smsmode_checkend:n`.)

20.2 Inheritance

1365 `<@@=stex_importmodule>`

`\stex_import_module_uri:nn`

```

1366 \cs_new_protected:Nn \stex_import_module_uri:nn {
1367   \str_set:Nx \l__stex_importmodule_archive_str { #1 }
1368   \str_set:Nn \l__stex_importmodule_path_str { #2 }
1369   \str_if_empty:NT \l__stex_importmodule_archive_str {
1370     \prop_if_empty:NF \l_stex_current_repository_prop {
1371       \prop_get:NnN \l_stex_current_repository_prop { id } \l__stex_importmodule_archive_str
1372     }
1373   }
1374
1375   \exp_args:NNNo \seq_set_split:Nnn \l_tmpb_seq ? { \l__stex_importmodule_path_str }
1376   \seq_pop_right:NN \l_tmpb_seq \l__stex_importmodule_name_str
1377   \str_set:Nx \l__stex_importmodule_path_str { \seq_use:Nn \l_tmpb_seq ? }
1378
1379   \str_if_empty:NTF \l__stex_importmodule_archive_str {
1380     \stex_modules_current_namespace:
1381     \str_if_empty:NF \l__stex_importmodule_path_str {
1382       \str_set:Nx \l_stex_module_ns_str {
1383         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1384       }
1385     }
1386   }{
1387     \stex_require_repository:n \l__stex_importmodule_archive_str
1388     \prop_get:cnN { c_stex_mathhub\l__stex_importmodule_archive_str _manifest_prop } { ns }
1389     \l_stex_module_ns_str
1390     \str_if_empty:NF \l__stex_importmodule_path_str {
1391       \str_set:Nx \l_stex_module_ns_str {
1392         \l_stex_module_ns_str / \l__stex_importmodule_path_str
1393       }
1394     }
```

```

1394     }
1395   }
1396 }

```

(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 1397 \str_new:N \l__stex_importmodule_name_str
\l_stex_importmodule_path_str 1398 \str_new:N \l__stex_importmodule_archive_str
\l_stex_importmodule_file_str 1399 \str_new:N \l__stex_importmodule_path_str
1400 \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>}
1401 \cs_new_protected:Nn \stex_import_require_module:nnnn {
1402   \exp_args:Nx \stex_if_module_exists:nF { #1 ? #4 } {
1403
1404     % archive
1405     \str_set:Nx \l_tmpa_str { #2 }
1406     \str_if_empty:NTF \l_tmpa_str {
1407       \seq_set_eq:NN \l_tmpa_seq \g_stex_currentfile_seq
1408     } {
1409       \stex_path_from_string:Nn \l_tmpb_seq { \l_tmpa_str }
1410       \seq_concat:NNN \l_tmpa_seq \c_stex_mathhub_seq \l_tmpb_seq
1411       \seq_put_right:Nn \l_tmpa_seq { source }
1412     }
1413
1414     % path
1415     \str_set:Nx \l_tmpb_str { #3 }
1416     \str_if_empty:NTF \l_tmpb_str {
1417       \str_set:Nx \l_tmpa_str { \stex_path_to_string:N \l_tmpa_seq / #4 }
1418
1419       \ltx@ifpackageloaded{babel} {
1420         \exp_args:NNx \prop_get:NnNF \c_stex_language_abbrevs_prop
1421           { \language } \l_tmpb_str {
1422           \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1423         }
1424       } {
1425         \str_clear:N \l_tmpb_str
1426       }
1427
1428       \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1429       \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1430         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1431       }{
1432         \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1433         \IfFileExists{ \l_tmpa_str.tex }{
1434           \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1435         }{
1436           % try english as default
1437           \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1438           \IfFileExists{ \l_tmpa_str.en.tex }{
1439             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }

```

```

1440         }{
1441             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1442         }
1443     }
1444 }
1445
1446 } {
1447     \seq_set_split:NnV \l_tmpb_seq / \l_tmpb_str
1448     \seq_concat:NNN \l_tmpa_seq \l_tmpa_seq \l_tmpb_seq
1449
1450     \ltx@ifpackageloaded{babel} {
1451         \exp_args:NnX \prop_get:NnNF \c_stex_language_abbrevs_prop
1452             { \language } \l_tmpb_str {
1453             \msg_error:nnn{stex}{error/unknownlanguage}{\language}
1454         }
1455     } {
1456         \str_clear:N \l_tmpb_str
1457     }
1458
1459     \stex_path_to_string:NN \l_tmpa_seq \l_tmpa_str
1460
1461     \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.\l_tmpb_str.tex}
1462     \IfFileExists{ \l_tmpa_str/#4.\l_tmpb_str.tex }{
1463         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.\l_tmpb_str.tex }
1464     }{
1465         \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.tex}
1466         \IfFileExists{ \l_tmpa_str/#4.tex }{
1467             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.tex }
1468         }{
1469             % try english as default
1470             \stex_debug:nn{modules}{Checking~\l_tmpa_str/#4.en.tex}
1471             \IfFileExists{ \l_tmpa_str/#4.en.tex }{
1472                 \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str/#4.en.tex }
1473             }{
1474                 \stex_debug:nn{modules}{Checking~\l_tmpa_str.\l_tmpb_str.tex}
1475                 \IfFileExists{ \l_tmpa_str.\l_tmpb_str.tex }{
1476                     \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.\l_tmpb_str.tex }
1477                 }{
1478                     \stex_debug:nn{modules}{Checking~\l_tmpa_str.tex}
1479                     \IfFileExists{ \l_tmpa_str.tex }{
1480                         \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.tex }
1481                     }{
1482                         % try english as default
1483                         \stex_debug:nn{modules}{Checking~\l_tmpa_str.en.tex}
1484                         \IfFileExists{ \l_tmpa_str.en.tex }{
1485                             \str_gset:Nx \g__stex_importmodule_file_str { \l_tmpa_str.en.tex }
1486                         }{
1487                             \msg_error:nnn{stex}{error/unknownmodule}{#1?#4}
1488                         }
1489                     }
1490                 }
1491             }
1492         }
1493     }

```

```

1494     }
1495
1496     \seq_set_eq:NN \l_tmpa_seq \g_stex_modules_in_file_seq
1497     \seq_clear:N \g_stex_modules_in_file_seq
1498     % \exp_args:Nnx \use:nn {
1499     \exp_args:No \stex_in_smsmode:nn { \g__stex_importmodule_file_str } {
1500     \seq_clear:N \l_stex_all_modules_seq
1501     \prop_clear:N \l_stex_current_module_prop
1502     \str_set:Nx \l_tmpb_str { #2 }
1503     \str_if_empty:NF \l_tmpb_str {
1504     \stex_set_current_repository:n { #2 }
1505     }
1506     \stex_debug:nn{modules}{Loading~\g__stex_importmodule_file_str}
1507     \input { \g__stex_importmodule_file_str }
1508     }
1509     % }{
1510
1511     % }
1512     \prop_gput:Noo \g_stex_module_files_prop
1513     \g__stex_importmodule_file_str \g_stex_modules_in_file_seq
1514     \seq_set_eq:NN \g_stex_modules_in_file_seq \l_tmpa_seq
1515
1516     \stex_if_module_exists:nF { #1 ? #4 } {
1517     \msg_error:nnn{stex}{error/unknownmodule}{
1518     #1?#4~(in~file~\g__stex_importmodule_file_str)
1519     }
1520     }
1521     }
1522     \stex_activate_module:n { #1 ? #4 }
1523     }

```

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

`\importmodule`

```

1524 \NewDocumentCommand \importmodule { O{} m } {
1525     \stex_import_module_uri:nn { #1 } { #2 }
1526     \stex_debug:nn{modules}{Importing~module:~
1527     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1528     }
1529     \stex_if_smsmode:F {
1530     \stex_import_require_module:nnnn
1531     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1532     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1533     \stex_annotate_invisible:nnn
1534     {import} { \l_stex_module_ns_str ? \l__stex_importmodule_name_str } {}
1535     }
1536     \exp_args:Nx \stex_add_to_current_module:n {
1537     \stex_import_require_module:nnnn
1538     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1539     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1540     }
1541     \exp_args:Nx \stex_add_import_to_current_module:n {
1542     \l_stex_module_ns_str ? \l__stex_importmodule_name_str
1543     }

```

```

1544 \stex_smsmode_set_codes:
1545 }
1546 \stex_deactivate_macro:Nn \importmodule {module-environments}

```

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

\usemodule

```

1547 \NewDocumentCommand \usemodule { 0{} m } {
1548   \stex_if_smsmode:F {
1549     \stex_import_module_uri:nn { #1 } { #2 }
1550     \stex_import_require_module:nnnn
1551     { \l_stex_module_ns_str } { \l__stex_importmodule_archive_str }
1552     { \l__stex_importmodule_path_str } { \l__stex_importmodule_name_str }
1553     \stex_annotate_invisible:nnn
1554     {usemodule} {\l_stex_module_ns_str ? \l__stex_importmodule_name_str} {}
1555   }
1556   \stex_smsmode_set_codes:
1557 }

```

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

```

1558 \endpackage

```

Chapter 21

STEX -Symbols Implementation

```
1559 <*package>
1560
1561 %%%%%%%%%%%%% symbols.dtx %%%%%%%%%%%%%
1562
Warnings and error messages
1563
```

21.1 Symbol Declarations

```
1564 <@@=stex_symdecl>

\l_stex_all_symbols_seq Stores all available symbols
1565 \seq_new:N \l_stex_all_symbols_seq

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

\STEXsymbol

1566 \NewDocumentCommand \STEXsymbol { m } {
1567   \stex_get_symbol:n { #1 }
1568   \exp_args:No
1569   \stex_invoke_symbol:n { \l_stex_get_symbol_uri_str }
1570 }

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

1571 \keys_define:nn { stex / symdecl } {
1572   name      .str_set_x:N = \l_stex_symdecl_name_str ,
1573   local     .bool_set:N = \l_stex_symdecl_local_bool ,
1574   args      .str_set_x:N = \l_stex_symdecl_args_str ,
1575   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
1576   align     .str_set:N   = \l_stex_symdecl_align_str , % TODO(?)
1577   gfc       .str_set:N   = \l_stex_symdecl_gfc_str , % TODO(?)
1578   specializes .str_set:N = \l_stex_symdecl_specializes_str , % TODO(?)
1579   def       .tl_set:N   = \l_stex_symdecl_definiens_tl
1580 }
```

```

1581
1582 \bool_new:N \l_stex_symdecl_make_macro_bool
1583
1584 \cs_new_protected:Nn \__stex_symdecl_args:n {
1585   \str_clear:N \l_stex_symdecl_name_str
1586   \str_clear:N \l_stex_symdecl_args_str
1587   \bool_set_false:N \l_stex_symdecl_local_bool
1588   \tl_clear:N \l_stex_symdecl_type_tl
1589   \tl_clear:N \l_stex_symdecl_definiens_tl
1590
1591   \keys_set:nn { stex / symdecl } { #1 }
1592 }

```

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

```

1593
1594 \NewDocumentCommand \symdecl { s O{} m } {
1595   \__stex_symdecl_args:n { #2 }
1596   \IfBooleanTF #1 {
1597     \bool_set_false:N \l_stex_symdecl_make_macro_bool
1598   } {
1599     \bool_set_true:N \l_stex_symdecl_make_macro_bool
1600   }
1601   \stex_symdecl_do:n { #3 }
1602   \stex_smsmode_set_codes:
1603 }
1604 \stex_deactivate_macro:Nn \symdecl {module-environments}

```

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

\stex_symdecl_do:n

```

1605 \cs_new_protected:Nn \stex_symdecl_do:n {
1606   \stex_if_in_module:F {
1607     % TODO throw error? some default namespace?
1608   }
1609
1610   \str_if_empty:NT \l_stex_symdecl_name_str {
1611     \str_set:Nx \l_stex_symdecl_name_str { #1 }
1612   }
1613
1614   \prop_if_exist:cT { g_stex_symdecl_
1615     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1616     \prop_item:Nn \l_stex_current_module_prop {name} ?
1617     \l_stex_symdecl_name_str
1618     _prop
1619   }{
1620     % TODO throw error (beware of circular dependencies)
1621   }
1622
1623   \prop_clear:N \l_tmpa_prop
1624   \prop_put:Nnx \l_tmpa_prop { module } {
1625     \prop_item:Nn \l_stex_current_module_prop {ns} ?
1626     \prop_item:Nn \l_stex_current_module_prop {name}
1627   }

```

```

1628 \seq_clear:N \l_tmpa_seq
1629 \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
1630 \prop_put:Nno \l_tmpa_prop { name } \l_stex_symdecl_name_str
1631 \prop_put:Nno \l_tmpa_prop { local } \l_stex_symdecl_local_bool
1632 \prop_put:Nno \l_tmpa_prop { type } \l_stex_symdecl_type_tl
1633
1634 \exp_args:No \stex_add_constant_to_current_module:n {
1635   \l_stex_symdecl_name_str
1636 }
1637
1638 % arity/args
1639 \int_zero:N \l_tmpb_int
1640
1641 \bool_set_true:N \l_tmpa_bool
1642 \str_map_inline:Nn \l_stex_symdecl_args_str {
1643   \token_case_meaning:NnF ##1 {
1644     0 {} 1 {} 2 {} 3 {} 4 {} 5 {} 6 {} 7 {} 8 {} 9 {}
1645     {\tl_to_str:n i} { \bool_set_false:N \l_tmpa_bool }
1646     {\tl_to_str:n b} { \bool_set_false:N \l_tmpa_bool }
1647     {\tl_to_str:n a} {
1648       \bool_set_false:N \l_tmpa_bool
1649       \int_incr:N \l_tmpb_int
1650     }
1651     {\tl_to_str:n B} {
1652       \bool_set_false:N \l_tmpa_bool
1653       \int_incr:N \l_tmpb_int
1654     }
1655   }{
1656     \msg_set:nnn{stex}{error/wrongargs}{
1657       args~value~in~symbol~declaration~for~
1658       \prop_item:Nn \l_stex_current_module_prop {ns} ?
1659       \prop_item:Nn \l_stex_current_module_prop {name} ?
1660       \l_stex_symdecl_name_str ~
1661       needs~to~be~
1662       i,~a,~b~or~B,~but~##1~given
1663     }
1664     \msg_error:nn{stex}{error/wrongargs}
1665   }
1666 }
1667 \bool_if:NTF \l_tmpa_bool {
1668   % possibly numeric
1669   \str_if_empty:NTF \l_stex_symdecl_args_str {
1670     \prop_put:Nnn \l_tmpa_prop { args } {}
1671     \prop_put:Nnn \l_tmpa_prop { arity } { 0 }
1672   }{
1673     \int_set:Nn \l_tmpa_int { \l_stex_symdecl_args_str }
1674     \prop_put:Nnx \l_tmpa_prop { arity } { \int_use:N \l_tmpa_int }
1675     \str_clear:N \l_tmpa_str
1676     \int_step_inline:nn \l_tmpa_int {
1677       \str_put_right:Nn \l_tmpa_str i
1678     }
1679     \prop_put:Nnx \l_tmpa_prop { args } { \l_tmpa_str }
1680   }
1681 } {

```



```

1682     \prop_put:Nnx \l_tmpa_prop { args } { \l_stex_symdecl_args_str }
1683     \prop_put:Nnx \l_tmpa_prop { arity }
1684       { \str_count:N \l_stex_symdecl_args_str }
1685   }
1686   \prop_put:Nnx \l_tmpa_prop { assocs } { \int_use:N \l_tmpb_int }
1687
1688
1689   % semantic macro
1690
1691   \bool_if:NT \l_stex_symdecl_make_macro_bool {
1692     \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1693       \prop_item:Nn \l_tmpa_prop { module } ?
1694       \prop_item:Nn \l_tmpa_prop { name }
1695     } }
1696
1697     \bool_if:NF \l_stex_symdecl_local_bool {
1698       \exp_args:Nx \stex_add_to_current_module:n {
1699         \tl_set:cx { #1 } { \stex_invoke_symbol:n {
1700           \prop_item:Nn \l_tmpa_prop { module } ?
1701           \prop_item:Nn \l_tmpa_prop { name }
1702         } }
1703       }
1704     }
1705   }
1706
1707   % add to all symbols
1708
1709   \bool_if:NF \l_stex_symdecl_local_bool {
1710     \exp_args:Nx \stex_add_to_current_module:n {
1711       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1712         \prop_item:Nn \l_tmpa_prop { module } ?
1713         \prop_item:Nn \l_tmpa_prop { name }
1714       }
1715     }
1716   }
1717
1718   \stex_debug:nn{symbols}{New~symbol:~
1719     \prop_item:Nn \l_tmpa_prop { module } ?
1720     \prop_item:Nn \l_tmpa_prop { name } ^^J
1721     Type:~\exp_not:o { \l_stex_symdecl_type_tl } ^^J
1722     Args:~\prop_item:Nn \l_tmpa_prop { args }
1723   }
1724
1725   % circular dependencies require this:
1726
1727   \prop_if_exist:cF {
1728     g_stex_symdecl_
1729     \prop_item:Nn \l_tmpa_prop { module } ?
1730     \prop_item:Nn \l_tmpa_prop { name }
1731     _prop
1732   } {
1733     \prop_gset_eq:cN {
1734       g_stex_symdecl_
1735       \prop_item:Nn \l_tmpa_prop { module } ?

```

```

1736     \prop_item:Nn \l_tmpa_prop { name }
1737     _prop
1738   } \l_tmpa_prop
1739 }
1740
1741 \stex_if_smsmode:TF {
1742   \bool_if:NF \l_stex_symdecl_local_bool {
1743     \exp_args:Nx \stex_add_to_sms:n {
1744       \prop_gset_from_keyval:cn {
1745         g_stex_symdecl_
1746         \prop_item:Nn \l_tmpa_prop { module } ?
1747         \prop_item:Nn \l_tmpa_prop { name }
1748         _prop
1749       } {
1750         name      = \prop_item:Nn \l_tmpa_prop { name }      ,
1751         module    = \prop_item:Nn \l_tmpa_prop { module }    ,
1752         notations = \prop_item:Nn \l_tmpa_prop { notations } ,
1753         local     = \prop_item:Nn \l_tmpa_prop { local }     ,
1754         type      = \prop_item:Nn \l_tmpa_prop { type }      ,
1755         args      = \prop_item:Nn \l_tmpa_prop { args }      ,
1756         arity     = \prop_item:Nn \l_tmpa_prop { arity }     ,
1757         assocs    = \prop_item:Nn \l_tmpa_prop { assocs }
1758       }
1759       \seq_put_right:Nn \exp_not:N \l_stex_all_symbols_seq {
1760         \prop_item:Nn \l_tmpa_prop { module } ?
1761         \prop_item:Nn \l_tmpa_prop { name }
1762       }
1763     }
1764   }
1765 }{
1766   \exp_args:NNx \seq_put_right:Nn \l_stex_all_symbols_seq {
1767     \prop_item:Nn \l_tmpa_prop { module } ?
1768     \prop_item:Nn \l_tmpa_prop { name }
1769   }
1770   \stex_if_do_html:T {
1771     \stex_annotate_invisible:nnn {symdecl} {
1772       \prop_item:Nn \l_tmpa_prop { module } ?
1773       \prop_item:Nn \l_tmpa_prop { name }
1774     } {
1775       \stex_annotate_invisible:nnn{type}{}{\l_stex_symdecl_type_tl$}
1776       \stex_annotate_invisible:nnn{args}{}{
1777         \prop_item:Nn \l_tmpa_prop { args }
1778       }
1779       \stex_annotate_invisible:nnn{macroname}{}{#1}
1780       \tl_if_empty:NF \l_stex_symdecl_definiens_tl {
1781         \stex_annotate_invisible:nnn{definiens}{}
1782         {\l_stex_symdecl_definiens_tl$}
1783       }
1784     }
1785   }
1786 }
1787 }

```

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

`\stex_get_symbol:n`

```
1788 \str_new:N \l_stex_get_symbol_uri_str
1789
1790 \cs_new_protected:Nn \stex_get_symbol:n {
1791   \tl_if_head_eq_catcode:nNTF { #1 } \relax {
1792     \__stex_symdecl_get_symbol_from_cs:n { #1 }
1793   }{
1794     % argument is a string
1795     % is it a command name?
1796     \cs_if_exist:cTF { #1 }{
1797       \cs_set_eq:Nc \l_tmpa_tl { #1 }
1798       \str_set:Nx \l_tmpa_str { \cs_argument_spec:N \l_tmpa_tl }
1799       \str_if_empty:NTF \l_tmpa_str {
1800         \exp_args:Nx \cs_if_eq:NNTF {
1801           \tl_head:N \l_tmpa_tl
1802         } \stex_invoke_symbol:n {
1803           \exp_args:No \__stex_symdecl_get_symbol_from_cs:n { \use:c { #1 } }
1804         }{
1805           \__stex_symdecl_get_symbol_from_string:n { #1 }
1806         }
1807       } {
1808         \__stex_symdecl_get_symbol_from_string:n { #1 }
1809       }
1810     }{
1811       % argument is not a command name
1812       \__stex_symdecl_get_symbol_from_string:n { #1 }
1813       % \l_stex_all_symbols_seq
1814     }
1815   }
1816 }
1817
1818 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_string:n {
1819   \str_set:Nn \l_tmpa_str { #1 }
1820   \bool_set_false:N \l_tmpa_bool
1821   \stex_if_in_module:T {
1822     \prop_get:NnN \l_stex_current_module_prop
1823     { constants } \l_tmpa_seq
1824     \exp_args:NNo \seq_if_in:NnT \l_tmpa_seq { \l_tmpa_str } {
1825       \bool_set_true:N \l_tmpa_bool
1826       \str_set:Nx \l_stex_get_symbol_uri_str {
1827         \prop_item:Nn \l_stex_current_module_prop { ns } ?
1828         \prop_item:Nn \l_stex_current_module_prop { name } ? #1
1829       }
1830     }
1831   }
1832   \bool_if:NF \l_tmpa_bool {
1833     \tl_set:Nn \l_tmpa_tl {
1834       \msg_set:nnn{stex}{error/unknownsymbol}{
1835         No~symbol~#1~found!
1836       }
1837     }
1838     \msg_error:nn{stex}{error/unknownsymbol}
1839   }
1840   \str_set:Nn \l_tmpa_str { #1 }
1841   \int_set:Nn \l_tmpa_int { \str_count:N \l_tmpa_str }
```

```

1841 \seq_map_inline:Nn \l_stex_all_symbols_seq {
1842   \str_set:Nn \l_tmpb_str { ##1 }
1843   \str_if_eq:eeT { \l_tmpa_str } {
1844     \str_range:Nnn \l_tmpb_str { -\l_tmpa_int } { -1 }
1845   } {
1846     \seq_map_break:n {
1847       \tl_set:Nn \l_tmpa_tl {
1848         \str_set:Nn \l_stex_get_symbol_uri_str {
1849           ##1
1850         }
1851       }
1852     }
1853   }
1854 }
1855 \l_tmpa_tl
1856 }
1857 }
1858
1859 \cs_new_protected:Nn \__stex_symdecl_get_symbol_from_cs:n {
1860   \exp_args:NNx \tl_set:Nn \l_tmpa_tl
1861     { \tl_tail:N \l_tmpa_tl }
1862   \tl_if_single:NTF \l_tmpa_tl {
1863     \exp_args:No \tl_if_head_is_group:nTF \l_tmpa_tl {
1864       \exp_after:wN \str_set:Nn \exp_after:wN
1865         \l_stex_get_symbol_uri_str \l_tmpa_tl
1866     }{
1867       % TODO
1868       % tail is not a single group
1869     }
1870   }{
1871     % TODO
1872     % tail is not a single group
1873   }
1874 }

```

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

21.2 Notations

```

1875 <@@=stex_notation>
      notation arguments:
1876 \keys_define:nn { stex / notation } {
1877   lang    .tl_set_x:N = \l__stex_notation_lang_str ,
1878   variant .tl_set_x:N = \l__stex_notation_variant_str ,
1879   prec    .tl_set_x:N = \l__stex_notation_prec_str ,
1880   op      .tl_set:N    = \l__stex_notation_op_tl ,
1881   unknown .code:n      = \str_set:Nx
1882     \l__stex_notation_variant_str \l_keys_key_str
1883 }
1884
1885 \cs_new_protected:Nn \__stex_notation_args:n {
1886   \str_clear:N \l__stex_notation_lang_str
1887   \str_clear:N \l__stex_notation_variant_str

```

```

1888 \str_clear:N \l__stex_notation_prec_str
1889 \tl_clear:N \l__stex_notation_op_tl
1890
1891 \keys_set:nn { stex / notation } { #1 }
1892 }

```

\notation

```

1893 \NewDocumentCommand \notation { 0{ } m } {
1894   \__stex_notation_args:n { #1 }
1895   \tl_clear:N \l_stex_symdecl_definiens_tl
1896   \stex_get_symbol:n { #2 }
1897   \stex_notation_do:nn { \l_stex_get_symbol_uri_str }
1898 }
1899 \stex_deactivate_macro:Nn \notation {module~environments}

```

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

\stex_notation_do:nn

```

1900 \cs_new_protected:Nn \stex_notation_do:nn {
1901   \prop_set_eq:Nc \l_tmpa_prop {
1902     g_stex_symdecl_ #1 _prop
1903   }
1904
1905   \prop_clear:N \l_tmpb_prop
1906   \prop_put:Nno \l_tmpb_prop { symbol } { #1 }
1907   \prop_put:Nno \l_tmpb_prop { language } \l__stex_notation_lang_str
1908   \prop_put:Nno \l_tmpb_prop { variant } \l__stex_notation_variant_str
1909
1910   % precedences
1911   \seq_clear:N \l_tmpb_seq
1912   \exp_args:NNno
1913   \str_if_empty:NTF \l__stex_notation_prec_str {
1914     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1915     \int_compare:nNnTF \l_tmpa_str = 0 {
1916       \exp_args:NNnx
1917       \prop_put:Nno \l_tmpb_prop { opprec }
1918       { \neginfprec }
1919     }{
1920       \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1921     }
1922   } {
1923     \str_if_eq:onTF \l__stex_notation_prec_str {nobrackets}{
1924       \exp_args:NNnx
1925       \prop_put:Nno \l_tmpb_prop { opprec }
1926       { \neginfprec }
1927       \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1928       \int_step_inline:nn { \l_tmpa_str } {
1929         \exp_args:NNx
1930         \seq_put_right:Nn \l_tmpb_seq { \infprec }
1931       }
1932     }{
1933       \seq_set_split:NnV \l_tmpa_seq ; \l__stex_notation_prec_str
1934       \seq_pop_left:NNTF \l_tmpa_seq \l_tmpa_str {
1935         \prop_put:Nno \l_tmpb_prop { opprec } \l_tmpa_str
1936         \seq_pop_left:NNT \l_tmpa_seq \l_tmpa_str {

```

```

1937         \exp_args:NNNo \exp_args:NNno \seq_set_split:Nnn
1938         \l_tmpa_seq {\tl_to_str:n{x} } { \l_tmpa_str }
1939         \seq_map_inline:Nn \l_tmpa_seq {
1940             \seq_put_right:Nn \l_tmpb_seq { ##1 }
1941         }
1942     }
1943     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1944 }{
1945     \prop_get:NnN \l_tmpa_prop { arity } \l_tmpa_str
1946     \int_compare:nNnTF \l_tmpa_str = 0 {
1947         \exp_args:NNnx
1948         \prop_put:Nno \l_tmpb_prop { opprec }
1949         { \infprec }
1950     }{
1951         \prop_put:Nnn \l_tmpb_prop { opprec } { 0 }
1952     }
1953 }
1954 }
1955 }
1956
1957 \seq_set_eq:NN \l_tmpa_seq \l_tmpb_seq
1958 \int_step_inline:nn { \l_tmpa_str } {
1959     \seq_pop_left:NNF \l_tmpa_seq \l_tmpb_str {
1960         \exp_args:NNx
1961         \seq_put_right:Nn \l_tmpb_seq {
1962             \prop_item:Nn \l_tmpb_prop { opprec }
1963         }
1964     }
1965 }
1966
1967 \prop_put:Nno \l_tmpb_prop { argprec } \l_tmpb_seq
1968 \tl_clear:N \l_tmpa_tl
1969
1970 \int_compare:nNnTF \l_tmpa_str = 0 {
1971     \exp_args:NNe
1972     \cs_set:Npn \l__stex_notation_macrocode_cs {
1973         \_stex_term_math_oms:nnnn { #1 }
1974         { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1975         { \prop_item:Nn \l_tmpb_prop { opprec } }
1976         { \exp_not:n { #2 } }
1977     }
1978     \__stex_notation_final:
1979 }{
1980     \prop_get:NnN \l_tmpa_prop { args } \l_tmpb_str
1981     \str_if_in:NnTF \l_tmpb_str b {
1982         \exp_args:Nne \use:nn
1983         {
1984             \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1985             \cs_set:Npn \l_tmpa_str { {
1986                 \_stex_term_math_omb:nnnn { #1 }
1987                 { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1988                 { \prop_item:Nn \l_tmpb_prop { opprec } }
1989                 { \exp_not:n { #2 } }
1990             }}

```

```

1991   }{
1992     \str_if_in:NnTF \l_tmpb_str B {
1993       \exp_args:Nne \use:nn
1994       {
1995         \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
1996         \cs_set:Npn \l_tmpa_str } { {
1997           \stex_term_math_omb:nnnn { #1 }
1998           { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
1999           { \prop_item:Nn \l_tmpb_prop { opprec } }
2000           { \exp_not:n { #2 } }
2001         } }
2002       }{
2003         \exp_args:Nne \use:nn
2004         {
2005           \cs_generate_from_arg_count:NNnn \l__stex_notation_macrocode_cs
2006           \cs_set:Npn \l_tmpa_str } { {
2007             \stex_term_math_oma:nnnn { #1 }
2008             { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }
2009             { \prop_item:Nn \l_tmpb_prop { opprec } }
2010             { \exp_not:n { #2 } }
2011           } }
2012         }
2013       }
2014
2015       \int_zero:N \l_tmpa_int
2016       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2017       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2018       \__stex_notation_arguments:
2019     }
2020   }

```

(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

```

2021 \cs_new_protected:Nn \__stex_notation_arguments: {
2022   \int_incr:N \l_tmpa_int
2023   \str_if_empty:NnTF \l_tmpa_str {
2024     \__stex_notation_final:
2025   }{
2026     \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2027     \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2028     \str_if_eq:NnTF \l_tmpb_str a {
2029       \__stex_notation_argument_assoc:n
2030     }{
2031       \str_if_eq:NnTF \l_tmpb_str B {
2032         \__stex_notation_argument_assoc:n
2033       }{
2034         \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2035         \tl_put_right:Nx \l_tmpa_tl {
2036           { \stex_term_math_arg:nnn
2037             { \int_use:N \l_tmpa_int }
2038             { \l_tmpb_str }
2039             { ####\int_use:N \l_tmpa_int }
2040           }

```

```

2041     }
2042     \__stex_notation_arguments:
2043   }
2044 }
2045 }
2046 }

```

(End definition for __stex_notation_arguments:.)

__stex_notation_argument_assoc:n

```

2047 \cs_new_protected:Nn \__stex_notation_argument_assoc:n {
2048   \seq_pop_left:NN \l_tmpa_seq \l_tmpb_str
2049   \cs_set:Npn \l_tmpa_cs ##1 ##2 { #1 }
2050   \tl_put_right:Nx \l_tmpa_tl {
2051     { \stex_term_math_assoc_arg:nnnn
2052       { \int_use:N \l_tmpa_int }
2053       { \l_tmpb_str }
2054       \exp_args:No \exp_not:n
2055       {\exp_after:wN { \l_tmpa_cs {####1} {####2} } }
2056       { ####\int_use:N \l_tmpa_int }
2057     }
2058   }
2059   \__stex_notation_arguments:
2060 }

```

(End definition for __stex_notation_argument_assoc:n.)

__stex_notation_final: Called after processing all notation arguments

```

2061 \cs_new_protected:Nn \__stex_notation_final: {
2062   \prop_get:NnN \l_tmpa_prop { arity } \l_tmpb_str
2063   \prop_get:NnN \l_tmpb_prop { symbol } \l_tmpa_str
2064   \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2065   \exp_args:Nne \use:nn
2066   {
2067     \cs_generate_from_arg_count:cNnn {
2068       stex_notation_ \l_tmpa_str \c_hash_str
2069       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2070       _cs
2071     }
2072     \cs_gset:Npn \l_tmpb_str } { {
2073       \exp_after:wN \exp_after:wN \exp_after:wN
2074       \exp_not:n \exp_after:wN \exp_after:wN \exp_after:wN
2075       { \exp_after:wN \l__stex_notation_macrocode_cs \l_tmpa_tl }
2076     } }
2077
2078   \tl_if_empty:NF \l__stex_notation_op_tl {
2079     \cs_gset:cpx {
2080       stex_op_notation_ \l_tmpa_str \c_hash_str
2081       \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2082       _cs
2083     } {
2084       \stex_term_oms:nnn {
2085         \l_tmpa_str \c_hash_str \l__stex_notation_variant_str \c_hash_str
2086         \l__stex_notation_lang_str

```



```

2087     }{
2088         \l_tmpa_str
2089     }{ \comp{ \exp_args:No \exp_not:n { \l__stex_notation_op_tl } } }
2090 }
2091 }
2092
2093
2094
2095 \stex_debug:nn{symbols}{
2096     Notation~\l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2097     ~for~\prop_item:Nn \l_tmpb_prop { symbol }^^J
2098     Operator~precedence:~
2099     \prop_item:Nn \l_tmpb_prop { opprec }^^J
2100     Argument~precedences:~
2101     \seq_use:Nn \l_tmpa_seq {,~}^^J
2102     Notation: \cs_meaning:c {
2103         stex_notation_ \l_tmpa_str \c_hash_str
2104         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2105         _cs
2106     }
2107 }
2108
2109 \prop_gset_eq:cN {
2110     g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2111     \c_hash_str \l__stex_notation_lang_str _prop
2112 } \l_tmpb_prop
2113
2114 \exp_args:Nx
2115 \stex_add_to_current_module:n {
2116     \prop_get:cnN {
2117         g_stex_symdecl_
2118         \prop_item:Nn \l_tmpb_prop { symbol }
2119         _prop
2120     } { notations } \exp_not:N \l_tmpa_seq
2121     \seq_put_right:Nn \exp_not:N \l_tmpa_seq {
2122         \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2123     }
2124     \prop_put:cno {
2125         g_stex_symdecl_
2126         \prop_item:Nn \l_tmpb_prop { symbol }
2127         _prop
2128     } { notations } \exp_not:N \l_tmpa_seq
2129 }
2130
2131 \stex_if_smsmode:TF {
2132     \stex_smsmode_set_codes:
2133     \exp_args:Nx \stex_add_to_sms:n {
2134         \prop_gset_from_keyval:cn {
2135             g_stex_notation_ \l_tmpa_str \c_hash_str \l__stex_notation_variant_str
2136             \c_hash_str \l__stex_notation_lang_str _prop
2137         } {
2138             symbol      = \prop_item:Nn \l_tmpb_prop { symbol }      ,
2139             language    = \prop_item:Nn \l_tmpb_prop { language }    ,
2140             variant     = \prop_item:Nn \l_tmpb_prop { variant }     ,

```

```

2141         opprec      = \prop_item:Nn \l_tmpb_prop { opprec }      ,
2142         argprec     = \prop_item:Nn \l_tmpb_prop { argprec }     ,
2143     }
2144 }
2145 }{
2146   \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2147   \seq_put_right:Nx \l_tmpa_seq {
2148     \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str
2149   }
2150   \prop_put:Nno \l_tmpa_prop { notations } \l_tmpa_seq
2151   \prop_set_eq:cN {
2152     g_stex_symdecl_ \l_tmpa_str _prop
2153   } \l_tmpa_prop
2154
2155   % HTML annotations
2156   \stex_if_do_html:T {
2157     \stex_annotate_invisible:nnn { notation }
2158     { \prop_item:Nn \l_tmpb_prop { symbol } } {
2159       \stex_annotate_invisible:nnn { notationfragment }
2160       { \l__stex_notation_variant_str \c_hash_str \l__stex_notation_lang_str }{}
2161       \prop_get:NnN \l_tmpb_prop { argprec } \l_tmpa_seq
2162       \stex_annotate_invisible:nnn { precedence }
2163       { \prop_item:Nn \l_tmpb_prop { opprec } ;
2164         \seq_use:Nn \l_tmpa_seq { x }
2165       }{}
2166
2167       \int_zero:N \l_tmpa_int
2168       \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2169       \tl_clear:N \l_tmpa_tl
2170       \int_step_inline:nn { \prop_item:Nn \l_tmpa_prop { arity } }{}{
2171         \int_incr:N \l_tmpa_int
2172         \str_set:Nx \l_tmpb_str { \str_head:N \l_tmpa_str }
2173         \str_set:Nx \l_tmpa_str { \str_tail:N \l_tmpa_str }
2174         \str_if_eq:VnTF \l_tmpb_str a {
2175           \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2176             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2177             \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2178           } }
2179         }{
2180           \str_if_eq:VnTF \l_tmpb_str B {
2181             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2182               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int a ,
2183               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int b
2184             } }
2185           }{
2186             \tl_set:Nx \l_tmpa_tl { \l_tmpa_tl {
2187               \c_hash_str \c_hash_str \int_use:N \l_tmpa_int
2188             } }
2189           }
2190         }
2191       }
2192       \stex_annotate_invisible:nnn { notationcomp }{}{
2193         $ \exp_args:Nno \use:nn { \use:c {
2194           stex_notation_ \prop_item:Nn \l_tmpb_prop { symbol }

```

```

2195         \c_hash_str \l__stex_notation_variant_str
2196         \c_hash_str \l__stex_notation_lang_str _cs
2197     } } { \l_tmpa_tl } $
2198 }
2199 }
2200 }
2201 }
2202 }

```

(End definition for `__stex_notation_final:`)

\symdef

```

2203 \keys_define:nn { stex / symdef } {
2204   name      .str_set:x:N = \l_stex_symdecl_name_str ,
2205   local     .bool_set:N = \l_stex_symdecl_local_bool ,
2206   args      .str_set:x:N = \l_stex_symdecl_args_str ,
2207   type      .tl_set:N   = \l_stex_symdecl_type_tl ,
2208   def       .tl_set:N   = \l_stex_symdecl_definiens_tl ,
2209   op        .tl_set:N   = \l__stex_notation_op_tl ,
2210   lang      .str_set:x:N = \l__stex_notation_lang_str ,
2211   variant   .str_set:x:N = \l__stex_notation_variant_str ,
2212   prec      .str_set:x:N = \l__stex_notation_prec_str ,
2213   unknown   .code:n     = \str_set:Nx
2214             \l__stex_notation_variant_str \l_keys_key_str
2215 }
2216
2217 \cs_new_protected:Nn \__stex_notation_symdef_args:n {
2218   \str_clear:N \l_stex_symdecl_name_str
2219   \str_clear:N \l_stex_symdecl_args_str
2220   \bool_set_false:N \l_stex_symdecl_local_bool
2221   \tl_clear:N \l_stex_symdecl_type_tl
2222   \tl_clear:N \l_stex_symdecl_definiens_tl
2223   \str_clear:N \l__stex_notation_lang_str
2224   \str_clear:N \l__stex_notation_variant_str
2225   \str_clear:N \l__stex_notation_prec_str
2226   \tl_clear:N \l__stex_notation_op_tl
2227
2228   \keys_set:nn { stex / symdef } { #1 }
2229 }
2230
2231 \NewDocumentCommand \symdef { 0{} m } {
2232   \__stex_notation_symdef_args:n { #1 }
2233   \bool_set_true:N \l_stex_symdecl_make_macro_bool
2234   \stex_symdecl_do:n { #2 }
2235   \exp_args:Nx \stex_notation_do:nn {
2236     \prop_item:Nn \l_tmpa_prop { module } ?
2237     \prop_item:Nn \l_tmpa_prop { name }
2238   }
2239 }
2240 \stex_deactivate_macro:Nn \symdef {module~environments}

```

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

```

2241 \endpackage

```

Chapter 22

STEX -Terms Implementation

```
2242 <*package>
2243
2244 %%%%%%%%%%% terms.dtx %%%%%%%%%%%
2245
2246 <@@=stex_terms>
2247
2248 Warnings and error messages
2249 \msg_new:nnn{stex}{error/nonotation}{
2250   Symbol~#1~invoked,~but~has~no~notation#2!
2251 }
2252 \msg_new:nnn{stex}{error/notationarg}{
2253   Error~in~parsing~notation~#1
2254 }
```

22.1 Symbol Invocations

Arguments:

```
2254 \keys_define:nn { stex / terms } {
2255   lang .tl_set_x:N = \l__stex_terms_lang_str ,
2256   variant .tl_set_x:N = \l__stex_terms_variant_str ,
2257   unknown .code:n = \str_set:Nx
2258     \l__stex_terms_variant_str \l_keys_key_str
2259 }
2260
2261 \cs_new_protected:Nn \__stex_terms_args:n {
2262   \str_clear:N \l__stex_terms_lang_str
2263   \str_clear:N \l__stex_terms_variant_str
2264   \str_clear:N \l__stex_terms_prec_str
2265   \tl_clear:N \l__stex_terms_op_tl
2266
2267   \keys_set:nn { stex / terms } { #1 }
2268 }
```

`\stex_invoke_symbol:n` Invokes a semantic macro

```

2269 \cs_new_protected:Nn \stex_invoke_symbol:n {
2270   \if_mode_math:
2271     \exp_after:wN \__stex_terms_invoke_math:n
2272   \else:
2273     \exp_after:wN \__stex_terms_invoke_text:n
2274   \fi: { #1 }
2275 }

```

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

`__stex_terms_invoke_math:n`

```

2276 \cs_new_protected:Nn \__stex_terms_invoke_math:n {
2277   \peek_charcode_remove:NTF ! {
2278     \peek_charcode:NTF [ {
2279       \__stex_terms_invoke_op:nw { #1 }
2280     }{
2281       \__stex_terms_invoke_op:nw { #1 } []
2282     }
2283   }{
2284     \peek_charcode_remove:NTF * {
2285       \__stex_terms_invoke_text:n { #1 }
2286     }{
2287       \peek_charcode:NTF [ {
2288         \__stex_terms_invoke_math:nw { #1 }
2289       }{
2290         \__stex_terms_invoke_math:nw { #1 } []
2291       }
2292     }
2293   }
2294 }

```

(End definition for `__stex_terms_invoke_math:n`.)

`__stex_terms_invoke_op:nw`

```

2295 \cs_new_protected:Npn \__stex_terms_invoke_op:nw #1 [#2] {
2296   \__stex_terms_args:n { #2 }
2297   \cs_if_exist:cTF {
2298     stex_op_notation_ #1 \c_hash_str
2299     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2300   }{
2301     \csname stex_op_notation_ #1 \c_hash_str
2302     \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str_cs
2303   \endcsname
2304   }{
2305     % TODO throw error
2306   }
2307 }

```

(End definition for `__stex_terms_invoke_op:nw`.)

`__stex_terms_invoke_math:nw`

```

2308 \cs_new_protected:Npn \__stex_terms_invoke_math:nw #1 [#2] {
2309   \__stex_terms_args:n { #2 }
2310   \prop_set_eq:Nc \l_tmpa_prop {
2311     g_stex_symdecl_ #1 _prop

```

```

2312 }
2313 \prop_get:NnN \l_tmpa_prop { notations } \l_tmpa_seq
2314 \seq_if_empty:NTF \l_tmpa_seq {
2315   \msg_error:nnnn{stex}{error/nonotation}{#1}{s}
2316 } {
2317   \seq_if_in:NxTF \l_tmpa_seq
2318   { \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str }{
2319     \use:c{
2320       stex_notation_ #1 \c_hash_str
2321       \l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2322       _cs
2323     }
2324   }{
2325     \str_if_empty:NTF \l__stex_terms_variant_str {
2326       \str_if_empty:NTF \l__stex_terms_lang_str {
2327         \seq_get_left:NN \l_tmpa_seq \l_tmpa_str
2328         \use:c{
2329           stex_notation_ #1 \c_hash_str \l_tmpa_str
2330           _cs
2331         }
2332       }{
2333         \msg_error:nn{stex}{error/nonotation}{#1}{
2334           ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2335         }
2336       }
2337     }{
2338       \msg_error:nn{stex}{error/nonotation}{#1}{
2339         ~\l__stex_terms_variant_str \c_hash_str \l__stex_terms_lang_str
2340       }
2341     }
2342   }
2343 }
2344 }

```

(End definition for `__stex_terms_invoke_math:nw`.)

`__stex_terms_invoke_text:n`

```

2345 \cs_new_protected:Nn \__stex_terms_invoke_text:n {
2346   \peek_charcode_remove:NTF ! {
2347     \stex_term_custom:nn { #1 } { }
2348   }{
2349     \prop_set_eq:Nc \l_tmpa_prop {
2350       g_stex_symdecl_ #1 _prop
2351     }
2352     \prop_get:NnN \l_tmpa_prop { args } \l_tmpa_str
2353     \exp_args:Nnx \stex_term_custom:nn { #1 } { \l_tmpa_str }
2354   }
2355 }

```

(End definition for `__stex_terms_invoke_text:n`.)

22.2 Terms

Precedences:

```

\infprec
\neginfprec
\l__stex_terms_downprec
2356 \tl_const:Nx \infprec {\int_use:N \c_max_int}
2357 \tl_const:Nx \neginfprec {-\int_use:N \c_max_int}
2358 \int_new:N \l__stex_terms_downprec
2359 \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
2360 \tl_set:Nn \l__stex_terms_left_bracket_str (
2361 \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

```

2362 \cs_new_protected:Nn \_stex_terms_maybe_brackets:nn {
2363 \int_compare:nNnTF { #1 } > \l__stex_terms_downprec {
2364 \bool_if:NTF \l_stex_inarray_bool { #2 }{
2365 \dobrackets { #2 }
2366 }
2367 }{ #2 }
2368 }

(End definition for \_stex_terms_maybe_brackets:nn.)

```

\dobrackets

```

2369 %\RequirePackage{scalerel}
2370 \cs_new_protected:Npn \dobrackets #1 {
2371 %\ThisStyle{\if D\m@switch
2372 % \exp_args:Nnx \use:nn
2373 % { \exp_after:wN \left\l__stex_terms_left_bracket_str #1 }
2374 % { \exp_not:N\right\l__stex_terms_right_bracket_str }
2375 % \else
2376 \exp_args:Nnx \use:nn
2377 { \l__stex_terms_left_bracket_str #1 }
2378 { \l__stex_terms_right_bracket_str }
2379 %\fi}
2380 }

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

```

\withbrackets

```

2381 \cs_new_protected:Npn \withbrackets #1 #2 #3 {
2382 \exp_args:Nnx \use:nn
2383 {
2384 \tl_set:Nx \l__stex_terms_left_bracket_str { #1 }
2385 \tl_set:Nx \l__stex_terms_right_bracket_str { #2 }
2386 #3
2387 }
2388 {
2389 \tl_set:Nn \exp_not:N \l__stex_terms_left_bracket_str
2390 {\l__stex_terms_left_bracket_str}
2391 \tl_set:Nn \exp_not:N \l__stex_terms_right_bracket_str

```

```

2392     {\l__stex_terms_right_bracket_str}
2393   }
2394 }

```

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

`\STEXinvisible`

```

2395 \cs_new_protected:Npn \STEXinvisible #1 {
2396   \stex_annotate_invisible:n { #1 }
2397 }

```

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

OMDoc terms:

`_stex_term_math_oms:nnnn`

```

2398 \cs_new_protected:Nn \_stex_term_oms:nnn {
2399   \stex_annotate:nnn{ OMID }{ #2 }{
2400     \stex_highlight_term:nn { #1 } { #3 }
2401   }
2402 }
2403
2404 \cs_new_protected:Nn \_stex_term_math_oms:nnnn {
2405   \__stex_terms_maybe_brackets:nn { #3 }{
2406     \_stex_term_oms:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2407   }
2408 }

```

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

`_stex_term_math_oma:nnnn`

```

2409 \cs_new_protected:Nn \_stex_term_oma:nnn {
2410   \stex_annotate:nnn{ OMA }{ #2 }{
2411     \stex_highlight_term:nn { #1 } { #3 }
2412   }
2413 }
2414
2415 \cs_new_protected:Nn \_stex_term_math_oma:nnnn {
2416   \__stex_terms_maybe_brackets:nn { #3 }{
2417     \_stex_term_oma:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2418   }
2419 }

```

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

`_stex_term_math_omb:nnnn`

```

2420 \cs_new_protected:Nn \_stex_term_ombind:nnn {
2421   \stex_annotate:nnn{ OMBIND }{ #2 }{
2422     \stex_highlight_term:nn { #1 } { #3 }
2423   }
2424 }
2425
2426 \cs_new_protected:Nn \_stex_term_math_omb:nnnn {
2427   \__stex_terms_maybe_brackets:nn { #3 }{
2428     \_stex_term_ombind:nnn { #1 } { #1\c_hash_str#2 } { #4 }
2429   }
2430 }

```


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

`_stex_term_math_arg:nnn`

```

2431 \cs_new_protected:Nn \_stex_term_arg:nn {
2432   \stex_unhighlight_term:n {
2433     \stex_annotate:nnn{ arg }{ #1 }{ #2 }
2434   }
2435 }
2436 \cs_new_protected:Nn \_stex_term_math_arg:nnn {
2437   \exp_args:Nnx \use:nn
2438     { \int_set:Nn \l__stex_terms_downprec { #2 }
2439       \_stex_term_arg:nn { #1 }{ #3 }
2440     }
2441   { \int_set:Nn \exp_not:N \l__stex_terms_downprec { \int_use:N \l__stex_terms_downprec }
2442   }

```

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

`_stex_term_math_assoc_arg:nnnn`

```

2443 \cs_new_protected:Nn \_stex_term_math_assoc_arg:nnnn {
2444   \seq_set_split:Nnn \l_tmpa_seq , { #4 }
2445   \int_compare:nNnTF { \seq_count:N \l_tmpa_seq } < 2 {
2446     \tl_set:Nn \l_tmpa_tl { #4 }
2447   }{
2448     \cs_set:Npn \l_tmpa_cs ##1 ##2 { #3 }
2449     \seq_reverse:N \l_tmpa_seq
2450     \seq_pop_left:NN \l_tmpa_seq \l_tmpb_tl
2451     \tl_set:No \l_tmpa_tl { \l_tmpb_tl }
2452
2453     \seq_map_inline:Nn \l_tmpa_seq {
2454       \exp_args:Nno \tl_set:No \l_tmpa_tl {
2455         \exp_args:Nno
2456           \l_tmpa_cs { ##1 } \l_tmpa_tl
2457       }
2458     }
2459
2460   }
2461   \exp_args:Nnno
2462   \_stex_term_math_arg:nnn{#1}{#2}\l_tmpa_tl
2463 }

```

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

`\stex_term_custom:nn`

```

2464 \cs_new_protected:Nn \stex_term_custom:nn {
2465   \str_set:Nn \l__stex_terms_custom_uri { #1 }
2466   \str_set:Nn \l_tmpa_str { #2 }
2467   \tl_clear:N \l_tmpa_tl
2468   \int_zero:N \l_tmpa_int
2469   \int_set:Nn \l_tmpb_int { \str_count:N \l_tmpa_str }
2470   \__stex_terms_custom_loop:
2471 }

```

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

_stex_terms_custom_loop:

```

2472 \cs_new_protected:Nn \_stex_terms_custom_loop: {
2473   \bool_set_false:N \l_tmpa_bool
2474   \bool_while_do:nn {
2475     \str_if_eq_p:ee X {
2476       \str_item:Nn \l_tmpa_str { \l_tmpa_int + 1 }
2477     }
2478   }{
2479     \int_incr:N \l_tmpa_int
2480   }
2481
2482   \peek_charcode:NTF [ {
2483     % notation/text component
2484     \_stex_terms_custom_component:w
2485   } {
2486     \int_compare:nNnTF \l_tmpa_int = \l_tmpb_int {
2487       % all arguments read => finish
2488       \_stex_terms_custom_final:
2489     } {
2490       % arguments missing
2491       \peek_charcode_remove:NTF * {
2492         % invisible, specific argument position or both
2493         \peek_charcode:NTF [ {
2494           % visible specific argument position
2495           \_stex_terms_custom_arg:wn
2496         } {
2497           % invisible
2498           \peek_charcode_remove:NTF * {
2499             % invisible specific argument position
2500             \_stex_terms_custom_arg_inv:wn
2501           } {
2502             % invisible next argument
2503             \_stex_terms_custom_arg_inv:wn [ \l_tmpa_int + 1 ]
2504           }
2505         }
2506       } {
2507         % next normal argument
2508         \_stex_terms_custom_arg:wn [ \l_tmpa_int + 1 ]
2509       }
2510     }
2511   }
2512 }

```

(End definition for _stex_terms_custom_loop:.)

_stex_terms_custom_arg_inv:wn

```

2513 \cs_new_protected:Npn \_stex_terms_custom_arg_inv:wn [ #1 ] #2 {
2514   \bool_set_true:N \l_tmpa_bool
2515   \_stex_terms_custom_arg:wn [ #1 ] { #2 }
2516 }

```

(End definition for _stex_terms_custom_arg_inv:wn.)

_stex_terms_custom_arg:wn

```

2517 \cs_new_protected:Npn \__stex_terms_custom_arg:wn [ #1 ] #2 {
2518   \str_set:Nx \l_tmpb_str {
2519     \str_item:Nn \l_tmpa_str { #1 }
2520   }
2521   \str_case:VnTF \l_tmpb_str {
2522     { X } {
2523       \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2524     }
2525     { i } { \__stex_terms_custom_set_X:n { #1 } }
2526     { b } { \__stex_terms_custom_set_X:n { #1 } }
2527     { a } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2528     { B } { \__stex_terms_custom_set_X:n { #1 } } % TODO ?
2529   }{}{
2530     \msg_error:nnn{stex}{error/notationarg}{\l__stex_terms_custom_uri}
2531   }
2532
2533   \bool_if:nTF \l_tmpa_bool {
2534     \tl_put_right:Nx \l_tmpa_tl {
2535       \stex_annotate_invisible:n {
2536         \stex_term_arg:nn { \int_eval:n { #1 } }
2537         \exp_not:n { { #2 } }
2538       }
2539     }
2540   } {
2541     \tl_put_right:Nx \l_tmpa_tl {
2542       \stex_term_arg:nn { \int_eval:n { #1 } }
2543       \exp_not:n { { #2 } }
2544     }
2545   }
2546
2547   \__stex_terms_custom_loop:
2548 }

```

(End definition for __stex_terms_custom_arg:wn.)

__stex_terms_custom_set_X:n

```

2549 \cs_new_protected:Nn \__stex_terms_custom_set_X:n {
2550   \str_set:Nx \l_tmpa_str {
2551     \str_range:Nnn \l_tmpa_str 1 { #1 - 1 }
2552     X
2553     \str_range:Nnn \l_tmpa_str { #1 + 1 } { -1 }
2554   }
2555 }

```

(End definition for __stex_terms_custom_set_X:n.)

__stex_terms_custom_component:

```

2556 \cs_new_protected:Npn \__stex_terms_custom_component:w [ #1 ] {
2557   \tl_put_right:Nn \l_tmpa_tl { \comp{ #1 } }
2558   \__stex_terms_custom_loop:
2559 }

```

(End definition for __stex_terms_custom_component:.)

`_stex_terms_custom_final:`

```

2560 \cs_new_protected:Nn \_stex_terms_custom_final: {
2561   \int_compare:nNnTF \l_tmpb_int = 0 {
2562     \exp_args:Nnno \_stex_term_oms:nnn
2563   }{
2564     \str_if_in:NnTF \l_tmpa_str {b} {
2565       \exp_args:Nnno \_stex_term_ombind:nnn
2566     } {
2567       \exp_args:Nnno \_stex_term_oma:nnn
2568     }
2569   }
2570   { \l__stex_terms_custom_uri } { \l__stex_terms_custom_uri } { \l_tmpa_tl }
2571 }

```

(End definition for `_stex_terms_custom_final:`.)

`\symref`

`\symname`

```

2572 \NewDocumentCommand \symref { m m }{
2573   \let\compemph_uri_prev:\compemph@uri
2574   \let\compemph@uri\symrefemph@uri
2575   \STEXsymbol{#1}![#2]
2576   \let\compemph@uri\compemph_uri_prev:
2577 }
2578
2579 \keys_define:nn { stex / symname } {
2580   post      .str_set_x:N      = \l_stex_symname_post_str
2581 }
2582
2583 \cs_new_protected:Nn \stex_symname_args:n {
2584   \str_clear:N \l_stex_symname_post_str
2585   \keys_set:nn { stex / symname } { #1 }
2586 }
2587
2588 \NewDocumentCommand \symname { 0{} m }{
2589   \stex_symname_args:n { #1 }
2590   \stex_get_symbol:n { #2 }
2591   \str_set:Nx \l_tmpa_str {
2592     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
2593   }
2594   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
2595
2596   \let\compemph_uri_prev:\compemph@uri
2597   \let\compemph@uri\symrefemph@uri
2598   \exp_args:NNx \use:nn
2599   \stex_invoke_symbol:n { { \l_stex_get_symbol_uri_str }![
2600     \l_tmpa_str \l_stex_symname_post_str
2601   ] }
2602   \let\compemph@uri\compemph_uri_prev:
2603 }

```

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

22.3 Notation Components

2604 $\langle @@=\text{stex_notationcomps} \rangle$

$\backslash\text{stex_highlight_term:nn}$

```

2605
2606 \str_new:N \l__stex_notationcomps_highlight_uri_str
2607 \cs_new_protected:Nn \stex_highlight_term:nn {
2608   \exp_args:Nnx
2609   \use:nn {
2610     \str_set:Nx \l__stex_notationcomps_highlight_uri_str { #1 }
2611     #2
2612   } {
2613     \str_set:Nx \exp_not:N \l__stex_notationcomps_highlight_uri_str
2614     { \l__stex_notationcomps_highlight_uri_str }
2615   }
2616 }
2617
2618 \cs_new_protected:Nn \stex_unhighlight_term:n {
2619   % \latexml_if:TF {
2620   %   #1
2621   % } {
2622   %   \scalatex_if:TF {
2623   %     #1
2624   %   } {
2625     #1 %\iffalse{{\fi}} #1 {{\iffalse}}\fi
2626   % }
2627   % }
2628 }
```

(End definition for $\backslash\text{stex_highlight_term:nn}$. This function is documented on page 29.)

```

\comp
\compemph@uri 2629 \cs_new_protected:Npn \comp #1 {
\compemph      2630   \str_if_empty:NF \l__stex_notationcomps_highlight_uri_str {
\defemph      2631     \scalatex_if:TF {
\defemph@uri  2632       \stex_annotate:nnn { comp }{ \l__stex_notationcomps_highlight_uri_str }{ #1 }
\symrefemph   2633     }{
\symrefemph@uri 2634       \exp_args:Nnx \compemph@uri { #1 } { \l__stex_notationcomps_highlight_uri_str }
2635     }
2636   }
2637 }
2638
2639 \cs_new_protected:Npn \compemph@uri #1 #2 {
2640   \compemph{ #1 }
2641 }
2642
2643
2644 \cs_new_protected:Npn \compemph #1 {
2645   \textcolor{blue}{#1}
2646 }
2647
2648 \cs_new_protected:Npn \defemph@uri #1 #2 {
2649   \defemph{#1}
2650 }
```

```

2651
2652 \cs_new_protected:Npn \defemph #1 {
2653   \textbf{#1}
2654 }
2655
2656 \cs_new_protected:Npn \symrefemph@uri #1 #2 {
2657   \symrefemph{#1}
2658 }
2659
2660 \cs_new_protected:Npn \symrefemph #1 {
2661   \textbf{#1}
2662 }

```

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

`\ellipses`

```

2663 \NewDocumentCommand \ellipses {} { \ldots }

```

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

```

\parray
\prmatrix
\parrayline
\parraylineh
\parraycell
2664 \bool_new:N \l_stex_inarray_bool
2665 \bool_set_false:N \l_stex_inarray_bool
2666 \NewDocumentCommand \parray { m m } {
2667   \begingroup
2668   \bool_set_true:N \l_stex_inarray_bool
2669   \begin{array}{#1}
2670     #2
2671   \end{array}
2672 \endgroup
2673 }
2674
2675 \NewDocumentCommand \prmatrix { m } {
2676   \begingroup
2677   \bool_set_true:N \l_stex_inarray_bool
2678   \begin{matrix}
2679     #1
2680   \end{matrix}
2681 \endgroup
2682 }
2683
2684 \def \parrayline #1 #2 {
2685   #1 #2 \bool_if:NT \l_stex_inarray_bool {\}
2686 }
2687
2688 \def \parraylineh #1 #2 {
2689   #1 #2 \bool_if:NT \l_stex_inarray_bool {\hline}
2690 }
2691
2692 \def \parraycell #1 {
2693   #1 \bool_if:NT \l_stex_inarray_bool {\&}
2694 }

```

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

```

2695 \endpackage

```

Chapter 23

STEX -Structural Features Implementation

```
2696 <*package>
2697
2698 %%%%%%%%%%% features.dtx %%%%%%%%%%%
2699
2700 <@@=stex_features>
      Warnings and error messages
2701
```

23.1 The feature environment

structural@feature

```
2702
2703 \NewDocumentEnvironment{structural@feature}{ m m m }{
2704   \stex_if_in_module:F {
2705     \msg_set:nnn{stex}{error/nomodule}{
2706       Structural~Feature~has~to~occur~in~a~module:\\
2707       Feature~#2~of~type~#1\\
2708       In~File:~\stex_path_to_string:N \g_stex_currentfile_seq
2709     }
2710     \msg_error:nn{stex}{error/nomodule}
2711   }
2712
2713   \str_set:Nx \l_stex_module_name_str {
2714     \prop_item:Nn \l_stex_current_module_prop
2715       { name } / #2 - feature
2716   }
2717
2718   \str_set:Nx \l_stex_module_ns_str {
2719     \prop_item:Nn \l_stex_current_module_prop
2720       { ns }
2721   }
2722
```

```

2723
2724 \str_clear:N \l_tmpa_str
2725 \seq_clear:N \l_tmpa_seq
2726 \tl_clear:N \l_tmpa_tl
2727 \exp_args:NNx \prop_set_from_keyval:Nn \l_stex_current_module_prop {
2728   origname = #2,
2729   name      = \l_stex_module_name_str ,
2730   ns        = \l_stex_module_ns_str ,
2731   imports   = \exp_not:o { \l_tmpa_seq } ,
2732   constants = \exp_not:o { \l_tmpa_seq } ,
2733   content   = \exp_not:o { \l_tmpa_tl } ,
2734   file      = \exp_not:o { \g_stex_currentfile_seq } ,
2735   lang      = \l_stex_module_lang_str ,
2736   sig       = \l_tmpa_str ,
2737   meta      = \l_tmpa_str ,
2738   feature   = #1 ,
2739 }
2740
2741 \stex_if_smsmode:TF {
2742   \stex_smsmode_set_codes:
2743 } {
2744   \begin{stex_annotate_env}{ feature:#1 }{}
2745   \stex_annotate_invisible:nnn{header}{}{ #3 }
2746 }
2747 }{
2748   \str_set:Nx \l_tmpa_str {
2749     c_stex_feature_
2750     \prop_item:Nn \l_stex_current_module_prop { ns } ?
2751     \prop_item:Nn \l_stex_current_module_prop { name }
2752     _prop
2753   }
2754   \prop_gset_eq:cN { \l_tmpa_str } \l_stex_current_module_prop
2755   \prop_gset_eq:NN \g_stex_last_feature_prop \l_stex_current_module_prop
2756   \stex_if_smsmode:TF {
2757     \exp_args:Nx \stex_add_to_sms:n {
2758       \prop_gset_from_keyval:cn {
2759         c_stex_feature_
2760         \prop_item:Nn \l_stex_current_module_prop { ns } ?
2761         \prop_item:Nn \l_stex_current_module_prop { name }
2762         _prop
2763       } {
2764         origname = #2,
2765         name      = \prop_item:cn { \l_tmpa_str } { name } ,
2766         ns        = \prop_item:cn { \l_tmpa_str } { ns } ,
2767         imports   = \prop_item:cn { \l_tmpa_str } { imports } ,
2768         constants = \prop_item:cn { \l_tmpa_str } { constants } ,
2769         content   = \prop_item:cn { \l_tmpa_str } { content } ,
2770         file      = \prop_item:cn { \l_tmpa_str } { file } ,
2771         lang      = \prop_item:cn { \l_tmpa_str } { lang } ,
2772         sig       = \prop_item:cn { \l_tmpa_str } { sig } ,
2773         meta      = \prop_item:cn { \l_tmpa_str } { meta } ,
2774         feature   = \prop_item:cn { \l_tmpa_str } { feature }
2775       }
2776     }

```



```

2777 } {
2778     \end{stex_annotate_env}
2779 }
2780 }
2781

```

23.2 Features

structure

```

2782
2783 \prop_new:N \l_stex_all_structures_prop
2784
2785 \keys_define:nn { stex / features / structure } {
2786     name .str_set_x:N = \l__stex_features_structure_name_str ,
2787 }
2788
2789 \cs_new_protected:Nn \__stex_features_structure_args:n {
2790     \str_clear:N \l__stex_features_structure_name_str
2791     \keys_set:nn { stex / features / structure } { #1 }
2792 }
2793
2794 %\stex_new_feature:nnnn { structure } { 0{ } m } {
2795 % \__stex_features_structure_args:n { ##1 }
2796 % \str_if_empty:NT \l__stex_features_structure_name_str {
2797 %     \str_set:Nx \l__stex_features_structure_name_str { ##2 }
2798 % }
2799 %} {
2800 %
2801 %}
2802
2803 \NewDocumentEnvironment{mathstructure}{ 0{ } m }{
2804     \__stex_features_structure_args:n { #1 }
2805     \str_if_empty:NT \l__stex_features_structure_name_str {
2806         \str_set:Nx \l__stex_features_structure_name_str { #2 }
2807     }
2808     \exp_args:Nnnx
2809     \begin{structural@feature}{ structure }
2810         { \l__stex_features_structure_name_str }{}
2811         \seq_clear:N \l_tmpa_seq
2812         \prop_put:Nno \l_stex_current_module_prop { fields } \l_tmpa_seq
2813     }{
2814         \prop_get:NnN \l_stex_current_module_prop { constants } \l_tmpa_seq
2815         \prop_get:NnN \l_stex_current_module_prop { fields } \l_tmpb_seq
2816         \str_set:Nx \l_tmpa_str {
2817             \prop_item:Nn \l_stex_current_module_prop { ns } ?
2818             \prop_item:Nn \l_stex_current_module_prop { name }
2819         }
2820         \seq_map_inline:Nn \l_tmpa_seq {
2821             \exp_args:NNx \seq_put_right:Nn \l_tmpb_seq { \l_tmpa_str ? ##1 }
2822         }
2823         \prop_put:Nno \l_stex_current_module_prop { fields } { \l_tmpb_seq }
2824         \exp_args:Nnx

```

```

2826 \AddToHookNext { env / mathstructure / after }{
2827 \symdecl[type = \exp_not:N\collection,def={\STEXsymbol{module-type}{
2828 \_stex_term_math_oms:nnnn { \l_tmpa_str }{}{0}{}}
2829 }}, name = \prop_item:Nn \l_stex_current_module_prop { origname }]{ #2 }
2830 \STEXexport {
2831 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2832 {\prop_item:Nn \l_stex_current_module_prop { origname }}
2833 {\l_tmpa_str}
2834 \prop_put:Nno \exp_not:N \l_stex_all_structures_prop
2835 {#2}{\l_tmpa_str}
2836 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2837 % \prop_item:Nn \l_stex_current_module_prop { origname },
2838 % \l_tmpa_str
2839 % }
2840 % \seq_put_right:Nn \exp_not:N \l_stex_all_structures_seq {
2841 % #2,\l_tmpa_str
2842 % }
2843 % \tl_set:cx { #2 } {
2844 % \stex_invoke_structure:n { \l_tmpa_str }
2845 % }
2846 % }
2847
2848 \end{structural@feature}
2849 % \g_stex_last_feature_prop
2850 }

```

\instantiate

```

2851 \seq_new:N \l__stex_features_structure_field_seq
2852 \str_new:N \l__stex_features_structure_field_str
2853 \str_new:N \l__stex_features_structure_def_tl
2854 \prop_new:N \l__stex_features_structure_prop
2855 \NewDocumentCommand \instantiate { m O{} m }{
2856 \stex_smsmode_set_codes:
2857 \prop_get:NnN \l_stex_all_structures_prop {#1} \l_tmpa_str
2858 \prop_set_eq:Nc \l__stex_features_structure_prop {
2859 c_stex_feature_\l_tmpa_str _prop
2860 }
2861 \seq_set_from_clist:Nn \l__stex_features_structure_field_seq { #2 }
2862 \seq_map_inline:Nn \l__stex_features_structure_field_seq {
2863 \seq_set_split:Nnn \l_tmpa_seq{=}{ ##1 }
2864 \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2865 \seq_get_left:NN \l_tmpa_seq \l_tmpa_tl
2866 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq
2867 {!} \l_tmpa_tl
2868 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {
2869 \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpb_seq 1}
2870 \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2871 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2872 }{
2873 \str_set:Nx \l__stex_features_structure_field_str \l_tmpa_tl
2874 \seq_get_right:NN \l_tmpa_seq \l_tmpa_tl
2875 \exp_args:NNno \seq_set_split:Nnn \l_tmpb_seq{!}
2876 \l_tmpa_tl
2877 \int_compare:nNnTF {\seq_count:N \l_tmpb_seq} > 1 {

```

```

2878         \seq_get_left:NN \l_tmpb_seq \l_tmpa_tl
2879         \seq_get_right:NN \l_tmpb_seq \l_tmpb_tl
2880     }{
2881         \tl_clear:N \l_tmpb_tl
2882     }
2883 }
2884 }{
2885     \seq_set_split:Nnn \l_tmpa_seq{!}{ ##1 }
2886     \int_compare:nNnTF {\seq_count:N \l_tmpa_seq} > 1 {
2887         \str_set:Nx \l__stex_features_structure_field_str {\seq_item:Nn \l_tmpa_seq 1}
2888         \seq_get_right:NN \l_tmpa_seq \l_tmpb_tl
2889         \tl_clear:N \l_tmpa_tl
2890     }{
2891         % TODO throw error
2892     }
2893 }
2894 % \l_tmpa_str: name
2895 % \l_tmpa_tl: definiens
2896 % \l_tmpb_tl: notation
2897 \tl_if_empty:NT \l__stex_features_structure_field_str {
2898     % TODO throw error
2899 }
2900 \str_clear:N \l_tmpb_str
2901
2902 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2903 \seq_map_inline:Nn \l_tmpa_seq {
2904     \seq_set_split:Nnn \l_tmpb_seq ? { ####1 }
2905     \seq_get_right:NN \l_tmpb_seq \l_tmpb_str
2906     \str_if_eq:NNT \l__stex_features_structure_field_str \l_tmpb_str {
2907         \seq_map_break:n {
2908             \str_set:Nn \l_tmpb_str { ####1 }
2909         }
2910     }
2911 }
2912 \prop_get:cnN { g_stex_symdecl_ \l_tmpb_str _prop } {args}
2913     \l_tmpb_str
2914
2915 \tl_if_empty:NTF \l_tmpb_tl {
2916     \tl_if_empty:NF \l_tmpa_tl {
2917         \exp_args:Nx \use:n {
2918             \symdecl[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2919         }
2920     }
2921 }{
2922     \tl_if_empty:NTF \l_tmpa_tl {
2923         \exp_args:Nx \use:n {
2924             \symdef[args=\l_tmpb_str]{#3/\l__stex_features_structure_field_str}\exp_after:wN\
2925         }
2926     }
2927 }{
2928     \exp_args:Nx \use:n {
2929         \symdef[args=\l_tmpb_str,def={\exp_args:No\exp_not:n{\l_tmpa_tl}}]{#3/\l__stex_fea
2930         \exp_after:wN\exp_not:n\exp_after:wN{\l_tmpb_tl}
2931     }

```

```

2932     }
2933   }
2934   % \par \prop_item:Nn \l_stex_current_module_prop {ns} ?
2935   % \prop_item:Nn \l_stex_current_module_prop {name} ?
2936   % #3/\l_stex_features_structure_field_str
2937   % \par
2938   % \expandafter\present\csname
2939   %   g_stex_symdecl_
2940   %   \prop_item:Nn \l_stex_current_module_prop {ns} ?
2941   %   \prop_item:Nn \l_stex_current_module_prop {name} ?
2942   %   #3/\l_stex_features_structure_field_str
2943   %   _prop
2944   % \endcsname
2945 }
2946
2947 \tl_clear:N \l__stex_features_structure_def_tl
2948
2949 \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
2950 \seq_map_inline:Nn \l_tmpa_seq {
2951   \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
2952   \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
2953   \exp_args:Nx \use:n {
2954     \tl_put_right:Nn \exp_not:N \l__stex_features_structure_def_tl {
2955
2956       }
2957   }
2958
2959   \prop_if_exist:cF {
2960     g_stex_symdecl_
2961     \prop_item:Nn \l_stex_current_module_prop {ns} ?
2962     \prop_item:Nn \l_stex_current_module_prop {name} ?
2963     #3/\l_tmpa_str
2964     _prop
2965   }{
2966     \prop_get:cnN { g_stex_symdecl_ ##1 _prop } {args}
2967     \l_tmpb_str
2968     \exp_args:Nx \use:n {
2969       \symdecl[args=\l_tmpb_str]{#3/\l_tmpa_str}
2970     }
2971   }
2972 }
2973
2974 \symdecl*[type={\STEXsymbol{module-type}}{
2975   \_stex_term_math_oms:nnnn {
2976     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2977     \prop_item:Nn \l__stex_features_structure_prop {name}
2978     }{}{0}{}
2979   }]}{#3}
2980
2981 % TODO: -> sms file
2982
2983 \tl_set:cx{ #3 }{
2984   \stex_invoke_structure:nnn {
2985     \prop_item:Nn \l_stex_current_module_prop {ns} ?

```

```

2986     \prop_item:Nn \l_stex_current_module_prop {name} ? #3
2987   } {
2988     \prop_item:Nn \l__stex_features_structure_prop {ns} ?
2989     \prop_item:Nn \l__stex_features_structure_prop {name}
2990   }
2991 }
2992
2993 }

```

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

\stex_invoke_structure:nnn

```

2994 % #1: URI of the instance
2995 % #2: URI of the instantiated module
2996 \cs_new_protected:Nn \stex_invoke_structure:nnn {
2997   \tl_if_empty:nTF{ #3 }{
2998     \prop_set_eq:Nc \l__stex_features_structure_prop {
2999       c_stex_feature_ #2 _prop
3000     }
3001     \tl_clear:N \l_tmpa_tl
3002     \prop_get:NnN \l__stex_features_structure_prop { fields } \l_tmpa_seq
3003     \seq_map_inline:Nn \l_tmpa_seq {
3004       \seq_set_split:Nnn \l_tmpb_seq ? { ##1 }
3005       \seq_get_right:NN \l_tmpb_seq \l_tmpa_str
3006       \cs_if_exist:cT {
3007         stex_notation_ #1/\l_tmpa_str \c_hash_str\c_hash_str _cs
3008       }{
3009         \tl_if_empty:NF \l_tmpa_tl {
3010           \tl_put_right:Nn \l_tmpa_tl {,}
3011         }
3012         \tl_put_right:Nx \l_tmpa_tl {
3013           \stex_invoke_symbol:n {#1/\l_tmpa_str}!
3014         }
3015       }
3016     }
3017     \exp_args:No \mathstrut \l_tmpa_tl
3018   }{
3019     \stex_invoke_symbol:n{#1/#3}
3020   }
3021 }

```

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

```

3022 </package>

```

Chapter 24

STEX -Statements Implementation

```
3023 <*package>
3024
3025 %%%%%%%%%%% features.dtx %%%%%%%%%%%
3026
3027 <@@=stex_statements>
    Warnings and error messages
3028
symboldoc
3029 \NewDocumentEnvironment{symboldoc}{m}{
3030   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3031   \seq_clear:N \l_tmpb_seq
3032   \seq_map_inline:Nn \l_tmpa_seq {
3033     \str_if_eq:nnF{ ##1 }{}{
3034       \stex_get_symbol:n { ##1 }
3035       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3036         \l_stex_get_symbol_uri_str
3037       }
3038     }
3039   }
3040   \par
3041   \exp_args:Nnnx
3042   \begin{stex_annotate_env}{symboldoc}{\seq_use:Nn \l_tmpb_seq {,}}
3043 }{
3044   \end{stex_annotate_env}
3045 }
3046 \seq_new:N \g_stex_statements_patched_seq
3047
3048 \cs_new_protected:Nn \stex_statements_set_patched:n {
3049   \seq_put_right:Nn \g_stex_statements_patched_seq {#1}
3050 }
3051
3052 \cs_new_protected:Nn \stex_statements_patch:nn {
3053   \seq_if_in:NnF \g_stex_statements_patched_seq {#1} {
```

```

3054 \AddToHook{begindocument}{
3055   \cs_if_exist:cTF{end#1}{
3056     \AddToHook{env/#1/before}[stex]{\use:c{__stex_statements_#2_begin:n}{}}
3057     \AddToHook{env/#1/after}[stex]{\use:c{__stex_statements_#2_end:}}
3058   }{
3059     \NewDocumentEnvironment{#1}{0{}}{
3060       \use:c{__stex_statements_#2_begin:n}{}
3061     }{
3062       \use:c{__stex_statements_#2_end:}
3063     }
3064   }
3065 }
3066 }
3067 }

```

24.1 Definitions

definition

```

3068
3069 \NewDocumentCommand \definiendum { 0{ } m m } {
3070   \stex_get_symbol:n { #2 }
3071   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3072   \scalatex_if:TF {
3073     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } { #3 }
3074   } {
3075     \exp_args:Nnx \defemph@uri { #3 } { \l_stex_get_symbol_uri_str }
3076   }
3077 }
3078 \stex_deactivate_macro:Nn \definiendum {definition~environments}
3079 \NewDocumentCommand \definame { 0{ } m } {
3080   % TODO: root
3081   \stex_get_symbol:n { #2 }
3082   \stex_ref_new_sym_target:n \l_stex_get_symbol_uri_str
3083   \str_set:Nx \l_tmpa_str {
3084     \prop_item:cn { g_stex_symdecl_ \l_stex_get_symbol_uri_str _prop } { name }
3085   }
3086   \exp_args:NNno \str_replace_all:Nnn \l_tmpa_str {-} {~}
3087   \scalatex_if:TF {
3088     \stex_annotate:nnn { definiendum } { \l_stex_get_symbol_uri_str } {
3089       \l_tmpa_str
3090     }
3091   } {
3092     \defemph@uri {
3093       \l_tmpa_str
3094     } { \l_stex_get_symbol_uri_str }
3095   }
3096 }
3097 \stex_deactivate_macro:Nn \definame {definition~environments}
3098
3099 \cs_new_protected:Nn \__stex_statements_defi_begin:n {
3100   \stex_reactivate_macro:N \definiendum
3101   \stex_reactivate_macro:N \definame
3102   \seq_set_split:Nnn \l_tmpa_seq , { #1 }

```

```

3103 \seq_clear:N \l_tmpb_seq
3104 \seq_map_inline:Nn \l_tmpa_seq {
3105   \str_if_eq:nnF{ ##1 }{}{
3106     \stex_get_symbol:n { ##1 }
3107     \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3108       \l_stex_get_symbol_uri_str
3109     }
3110   }
3111 }
3112 \stex_smsmode_set_codes:
3113 \exp_args:Nnnx
3114 \begin{stex_annotate_env}{definition}{\seq_use:Nn \l_tmpb_seq {,}}
3115 }
3116
3117 \cs_new_protected:Nn \__stex_statements_defi_end: {
3118   \end{stex_annotate_env}
3119 }

Hook:

3120 \stex_statements_patch:nn{definition}{defi}

```

24.2 Assertions

assertion

```

3121 \cs_new_protected:Nn \__stex_statements_assertion_begin:n {
3122   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3123   \seq_clear:N \l_tmpb_seq
3124   \seq_map_inline:Nn \l_tmpa_seq {
3125     \str_if_eq:nnF{ ##1 }{}{
3126       \stex_get_symbol:n { ##1 }
3127       \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3128         \l_stex_get_symbol_uri_str
3129       }
3130     }
3131   }
3132   \stex_smsmode_set_codes:
3133   \exp_args:Nnnx
3134   \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3135 }
3136
3137 \cs_new_protected:Nn \__stex_statements_assertion_end: {
3138   \end{stex_annotate_env}
3139 }

Hook:

3140 \stex_statements_patch:nn{assertion}{assertion}

```

theorem

```

3141 \cs_new_protected:Nn \__stex_statements_theorem_begin:n {
3142   \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3143   \seq_clear:N \l_tmpb_seq
3144   \seq_map_inline:Nn \l_tmpa_seq {

```



```

3145 \str_if_eq:nnF{ ##1 }{}{
3146 \stex_get_symbol:n { ##1 }
3147 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3148 \l_stex_get_symbol_uri_str
3149 }
3150 }
3151 }
3152 \stex_smsmode_set_codes:
3153 \exp_args:Nnnx
3154 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3155 }
3156
3157 \cs_new_protected:Nn \__stex_statements_theorem_end: {
3158 \end{stex_annotate_env}
3159 }

```

Hook:

```

3160 \stex_statements_patch:nn{theorem}{theorem}

```

lemma

```

3161 \cs_new_protected:Nn \__stex_statements_lemma_begin:n {
3162 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3163 \seq_clear:N \l_tmpb_seq
3164 \seq_map_inline:Nn \l_tmpa_seq {
3165 \str_if_eq:nnF{ ##1 }{}{
3166 \stex_get_symbol:n { ##1 }
3167 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3168 \l_stex_get_symbol_uri_str
3169 }
3170 }
3171 }
3172 \stex_smsmode_set_codes:
3173 \exp_args:Nnnx
3174 \begin{stex_annotate_env}{assertion}{\seq_use:Nn \l_tmpb_seq {,}}
3175 }
3176
3177 \cs_new_protected:Nn \__stex_statements_lemma_end: {
3178 \end{stex_annotate_env}
3179 }

```

Hook:

```

3180 \stex_statements_patch:nn{lemma}{lemma}

```

axiom

```

3181 \cs_new_protected:Nn \__stex_statements_axiom_begin:n {
3182 \seq_set_split:Nnn \l_tmpa_seq , { #1 }
3183 \seq_clear:N \l_tmpb_seq
3184 \seq_map_inline:Nn \l_tmpa_seq {
3185 \str_if_eq:nnF{ ##1 }{}{
3186 \stex_get_symbol:n { ##1 }
3187 \exp_args:NNo \seq_put_right:Nn \l_tmpb_seq {
3188 \l_stex_get_symbol_uri_str
3189 }

```

```

3190     }
3191   }
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_axiom_end: {
3198   \end{stex_annotate_env}
3199 }

```

Hook:

```

3200 \stex_statements_patch:nn{axiom}{axiom}

```

24.3 Examples

example

```

3201 \cs_new_protected:Nn \__stex_statements_example_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   \stex_smsmode_set_codes:
3213   \exp_args:Nnnx
3214   \begin{stex_annotate_env}{example}{\seq_use:Nn \l_tmpb_seq {,}}
3215 }
3216
3217 \cs_new_protected:Nn \__stex_statements_example_end: {
3218   \end{stex_annotate_env}
3219 }

```

Hook:

```

3220 \stex_statements_patch:nn{example}{example}

```

```

3221 \end{package}

```

Some auxiliary code, and clean up to be executed at the end of the package.

Chapter 25

STEX -Others Implementation

```
3222 <*package>
3223
3224 %%%%%%%%%% others.dtx %%%%%%%%%%
3225
3226 <@@=stex_others>
    Warnings and error messages
3227 % None

\MSC Math subject classifier

3228 \NewDocumentCommand \MSC {m} {
3229 % TODO
3230 }

(End definition for \MSC. This function is documented on page 10.)
    Patching tikzinput, if loaded
3231 \@ifpackageloaded{tikzinput}{
3232 \RequirePackage{stex-tikzinput}
3233 }{}
3234 </package>
```

Chapter 26

STEX -Metatheory Implementation

```
3235 <*package>
3236 <@@=stex_modules>
3237
3238 %%%%%%%%%%% metatheory.dtx %%%%%%%%%%%
3239
3240 \str_const:Nn \c_stex_metatheory_ns_str {http://mathhub.info/sTeX}
3241 \begingroup
3242 \stex_module_setup:nn{
3243   ns=\c_stex_metatheory_ns_str,
3244   meta=NONE
3245 }{Metatheory}
3246 \stex_reactivate_macro:N \symdecl
3247 \stex_reactivate_macro:N \notation
3248 \stex_reactivate_macro:N \symdef
3249 \ExplSyntaxOff
3250 \csname stex_suppress_html:n\endcsname{
3251   % is-a (a:A, a \in A, a is an A, etc.)
3252   \symdecl[args=ai]{isa}
3253   \notation[typed]{isa}{#1 \comp{:} #2}{#1 \comp, #2}
3254   \notation[in]{isa}{#1 \comp\in #2}{#1 \comp, #2}
3255   \notation[pred]{isa}{#2\comp(#1 \comp)}{#1 \comp, #2}
3256
3257   % bind (\forall, \Pi, \lambda etc.)
3258   \symdecl[args=Bi]{bind}
3259   \notation[forall]{bind}{\comp\forall #1.\;#2}{#1 \comp, #2}
3260   \notation[\Pi]{bind}{\comp\prod_{#1}#2}{#1 \comp, #2}
3261   \notation[deffun]{bind}{\comp( #1 \comp)\; \to\;}{#1 \comp, #2}
3262
3263   % dummy variable
3264   \symdecl{dummyvar}
3265   \notation[underscore]{dummyvar}{\comp\_}
3266   \notation[dot]{dummyvar}{\comp\cdot}
3267   \notation[dash]{dummyvar}{\comp{\rm --}}
3268
3269   %fromto (function space, Hom-set, implication etc.)
```

```

3270 \symdecl[args=ai]{fromto}
3271 \notation[xarrow]{fromto}{#1 \comp\to #2}{#1 \comp\times #2}
3272 \notation[arrow]{fromto}{#1 \comp\to #2}{#1 \comp\to #2}
3273
3274 % mapto (lambda etc.)
3275 %\symdecl[args=Bi]{mapto}
3276 %\notation[mapsto]{mapto}{#1 \comp\mapsto #2}{#1 \comp, #2}
3277 %\notation[lambda]{mapto}{\comp\lambda #1 \comp. \; #2}{#1 \comp, #2}
3278 %\notation[lambdau]{mapto}{\comp\lambda_{#1} \comp. \; #2}{#1 \comp, #2}
3279
3280 % function/operator application
3281 \symdecl[args=ia]{apply}
3282 \notation[prec=0;0x\neginfprec,parens]{apply}{#1 \comp( #2 \comp)}{#1 \comp, #2}
3283 \notation[prec=0;0x\neginfprec,lambda]{apply}{#1 \; #2 }{#1 \; #2}
3284
3285 % ‘‘type’’ of all collections (sets, classes, types, kinds)
3286 \symdecl{collection}
3287 \notation[U]{collection}{\comp{\mathcal{U}}{}}
3288 \notation[set]{collection}{\comp{\textsf{Set}}{}}
3289
3290 % sequences
3291 \symdecl[args=1]{seqtype}
3292 \notation[kleene]{seqtype}{#1^{\comp\ast}}
3293
3294 \symdef[args=2,li]{sequence-index}{#1_{#2}}
3295 \notation[ui]{sequence-index}{#1^{\#2}}
3296
3297 %\symdef[args=3,li]{sequence-from-to}{#1_{#2}\comp{\,\ellipses,}#1_{#3}}
3298 %\notation[ui]{sequence-from-to}{#1^{\#2}\comp{\,\ellipses,}#1^{\#3}}
3299 % ^ superceded by \aseqfromto and \livar/\uivar
3300
3301 \symdef[args=a,prec=nobrackets]{aseqdots}{#1\comp{\,\ellipses}}{#1\comp,#2}
3302 \symdef[args=ai,prec=nobrackets]{aseqfromto}{#1\comp{\,\ellipses\comp,}#2 }{#1\comp,#2}
3303 \symdef[args=aai,prec=nobrackets]{aseqfromtovia}{#1\comp{\,\ellipses\comp,}#2\comp{\,\ellipses\comp,}#3}{#1\comp,#2}
3304
3305 % letin (‘‘let’’, local definitions, variable substitution)
3306 \symdecl[args=bii]{letin}
3307 \notation[let]{letin}{\comp{\rm let}}{\;#1\comp{=}\;#2\; \comp{\rm in}}{\;#3}
3308 \notation[subst]{letin}{#3 \comp[ #1 \comp/ #2 \comp]}
3309 \notation[frac]{letin}{#3 \comp[ \frac{#2}{#1} \comp]}
3310
3311 % structures
3312 \symdecl*[args=1]{module-type}
3313 \notation{module-type}{\mathtt{MOD} #1}
3314 \symdecl[name=mathematical-structure,args=a]{mathstruct} % TODO
3315 \notation[angle,prec=nobrackets]{mathstruct}{\comp\angle #1 \comp\rangle}{#1 \comp, #2}
3316
3317 }
3318 \ExplSyntaxOn
3319 \stex_add_to_current_module:n{
3320   \let\nappa\apply
3321   \def\nappli#1#2#3#4{\apply{#1}{\naseqli{#2}{#3}{#4}}}
3322   \def\livar{\csname sequence-index\endcsname[li]}
3323   \def\uivar{\csname sequence-index\endcsname[ui]}

```

```

3324     \def\naseqli#1#2#3{\aseqfromto{\livar{#1}{#2}}{\livar{#1}{#3}}}
3325     \def\nasequi#1#2#3{\aseqfromto{\uivar{#1}{#2}}{\uivar{#1}{#3}}}
3326   }
3327   \__stex_modules_end_module:
3328   \endgroup
3329 </package>

```

Chapter 27

Tikzinput Implementation

```
3330 <*package>
3331
3332 %%%%%%%%%% tikzinput.dtx %%%%%%%%%%
3333
3334 \ProvidesExplPackage{tikzinput}{2021/08/31}{1.9}{bla}
3335 \RequirePackage{l3keys2e}
3336
3337 \keys_define:nn { tikzinput } {
3338   image .bool_set:N = \c_tikzinput_image_bool,
3339   image .default:n = false ,
3340 }
3341
3342 \ProcessKeysOptions { tikzinput }
3343
3344 \bool_if:NTF \c_tikzinput_image_bool {
3345   \RequirePackage{graphicx}
3346
3347   \providecommand\usetikzlibrary[]{}
3348   \newcommand\tikzinput[2][]{\includegraphics[#1]{#2}}
3349 }{
3350   \RequirePackage{tikz}
3351   \RequirePackage{standalone}
3352
3353   \newcommand \tikzinput [2] [] {
3354     \setkeys{Gin}{#1}
3355     \ifx \Gin@ewidth \Gin@exclamation
3356       \ifx \Gin@eheight \Gin@exclamation
3357         \input { #2 }
3358       \else
3359         \resizebox{!}{ \Gin@eheight }{
3360           \input { #2 }
3361         }
3362       \fi
3363     \else
3364       \ifx \Gin@eheight \Gin@exclamation
3365         \resizebox{ \Gin@ewidth }{!}{
3366           \input { #2 }
3367         }
3368       \fi
3369     \fi
3370   }
3371 }
```

```

3368         \else
3369             \resizebox{ \Gin@ewidth }{ \Gin@eheight }{
3370                 \input { #2 }
3371             }
3372         \fi
3373     \fi
3374 }
3375 }
3376
3377 \newcommand \ctikzinput [2] [] {
3378     \begin{center}
3379         \tikzinput [1] {#2}
3380     \end{center}
3381 }
3382
3383 \@ifpackageloaded{stex}{
3384     \RequirePackage{stex-tikzinput}
3385 }{}
3386
3387 </package>
3388 <*stex>
3389 \ProvidesExplPackage{stex-tikzinput}{2021/08/31}{1.9}{bla}
3390 \RequirePackage{stex}
3391 \RequirePackage{tikzinput}
3392
3393 \newcommand\mhtikzinput[2] [] {%
3394     \def\Gin@mhrepos{}\setkeys{Gin}{#1}%
3395     \stex_in_repository:nn\Gin@mhrepos{
3396         \tikzinput[#1]{\mhp{##1}{#2}}
3397     }
3398 }
3399 \newcommand\cmhtikzinput[2] [] {\begin{center}\mhtikzinput[#1]{#2}\end{center}}
3400 </stex>

```

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

Chapter 28

document-structure.sty Implementation

28.1 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.

```
3401 \*cls)
3402 \ProvidesExplClass{omdoc}{2020/10/19}{1.4}{OMDoc Documents}
3403 \RequirePackage{l3keys2e,expl-keystr-compatible}
```

28.2 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`

```
3404 \keys_define:nn{ document-structure / pkg }{
3405   class      .initial:n    = {article},
3406   class      .str_set_x:N   = \c_document_structure_class_str,
3407   minimal    .bool_set:N    = \c_document_structure_minimal_bool,
3408   report     .code:n        = {
3409     \ClassWarning{omdoc}{the option 'report' is deprecated, use 'class=report', instead}
3410     \str_set:Nn \c_document_structure_class_str {report}
3411   },
3412   book       .code:n        = {
3413     \ClassWarning{omdoc}{the option 'book' is deprecated, use 'class=book', instead}
3414     \str_set:Nn \c_document_structure_class_str {book}
3415   },
3416   bookpart   .code:n        = {
3417     \ClassWarning{omdoc}{the option 'bookpart' is deprecated, use 'class=book,topsect=chapter}
3418     \str_set:Nn \c_document_structure_class_str {book}
3419     \str_set:Nn \c_document_structure_topsect_str {chapter}
3420   },
```

```

3421 docopt      .str_set_x:N = \c_document_structure_dcopt_str,
3422 unknown     .code:n      = {
3423   \PassOptionsToPackage{ \CurrentOption }{ omdoc }
3424 }
3425 }
3426
3427 \ProcessKeysOptions{ document-structure / pkg }
3428 \exp_after:wN\LoadClass\exp_after:wN[\c_document_structure_dcopt_str]
3429   {\c_document_structure_class_str}
3430

```

28.3 Beefing up the document environment

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

```

3431 \RequirePackage{omdoc}
3432 \bool_if:NF \c_document_structure_minimal_bool {
3433   \RequirePackage{stex}

```

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.⁸

```

3434 \keys_define:nn { document-structure / document }{
3435   id .str_set_x:N = \c_document_structure_document_id_str
3436 }
3437 \let\__orig_document=\document
3438 \renewcommand{\document}[1][]{
3439   \keys_set:nn{ document-structure / document }{ #1 }
3440   \stex_ref_new_doc_target:n { \c_document_structure_document_id_str }
3441   \__orig_document
3442 }

```

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

```

3443 }
3444 \</cls>

```

28.4 Implementation: OMDoc Package

```

3445 \<*package>
3446 \ProvidesExplPackage{omdoc}{2020/10/19}{1.4}{OMDoc document Structure}
3447 \RequirePackage{expl-keystr-compat, l3keys2e}

```

28.5 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).

3448

⁸EdNOTE: faking documentkeys for now. @HANG, please implement

```

3449 \keys_define:nn{ document-structure / pkg }{
3450   class      .str_set_x:N = \c_document_structure_class_str,
3451   topsect    .str_set_x:N = \c_document_structure_topsect_str,
3452   % showignores .bool_set:N = \c_document_structure_showignores_bool,
3453 }
3454 \ProcessKeysOptions{ document-structure / pkg }
3455 \str_if_empty:NT \c_document_structure_class_str {
3456   \str_set:Nn \c_document_structure_class_str {article}
3457 }
3458 \str_if_empty:NT \c_document_structure_topsect_str {
3459   \str_set:Nn \c_document_structure_topsect_str {section}
3460 }

```

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

```

3461 \RequirePackage{xspace}
3462 \RequirePackage{comment}
3463 \@ifpackageloaded{babel}{\RequirePackage[base]{babel}}

```

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

```

3464 \@ifpackageloaded{babel}{
3465   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
3466   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
3467     \input{omdoc-ngerman.ldf}
3468   }
3469 }{}
3470 %\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.

```

3471 \int_new:N \l_document_structure_section_level_int
3472 \str_case:VnF \c_document_structure_topsect_str {
3473   {part}{
3474     \int_set:Nn \l_document_structure_section_level_int {0}
3475   }
3476   {chapter}{
3477     \int_set:Nn \l_document_structure_section_level_int {1}
3478   }
3479 }{
3480   \str_case:VnF \c_document_structure_class_str {
3481     {book}{
3482       \int_set:Nn \l_document_structure_section_level_int {0}
3483     }
3484     {report}{
3485       \int_set:Nn \l_document_structure_section_level_int {0}
3486     }
3487   }{
3488     \int_set:Nn \l_document_structure_section_level_int {2}
3489   }
3490 }

```

28.6 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.⁹

EdN:9

```
3491 \def\current@section@level{document}%
3492 \newcommand\currentsectionlevel{\lowercase\expandafter{\current@section@level}\xspace}%
3493 \newcommand\Currentsectionlevel{\expandafter\MakeUppercase\current@section@level\xspace}%
```

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

`\skipomgroup`

```
3494 \cs_new_protected:Npn \skipomgroup {
3495   \ifcase\l_document_structure_section_level_int
3496   \or\stepcounter{chapter}
3497   \or\stepcounter{section}
3498   \or\stepcounter{subsection}
3499   \or\stepcounter{subsubsection}
3500   \or\stepcounter{paragraph}
3501   \or\stepcounter{subparagraph}
3502   \fi
3503 }
```

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

`blindomgroup`

```
3504 \newcommand\at@begin@blindomgroup[1]{
3505   \newenvironment{blindomgroup}
3506   {
3507     \int_incr:N\l_document_structure_section_level_int
3508     \at@begin@blindomgroup\l_document_structure_section_level_int
3509   }{
3510     \int_decr:N\l_document_structure_section_level_int
3511   }
```

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

```
3512 \newcommand\omgroup@nonum[2]{
3513   \ifx\hyper@anchor\undefined\else\phantomsection\fi
3514   \addcontentsline{toc}{#1}{#2}\@nameuse{#1}*{#2}
3515 }
```

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

⁹EdNOTE: MK: we may have to experiment with the more powerful uppercasing macro from `mfirstuc.sty` once we internationalize.

`\omgroup@num` convenience macro: `\omgroup@nonum{<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.

```

3516 \newcommand\omgroup@num[2]{
3517   \tl_if_empty:NTF \l_@@_omgroup_short_tl {
3518     \@nameuse{#1}{#2}
3519   }{
3520     \cs_if_exist:NTF\rdfmata@sectioning{
3521       \@nameuse{#1}[\l_@@_omgroup_short_tl]{#2}
3522     }{
3523       \@nameuse{rdfmata@#1old}[\l_@@_omgroup_short_tl]{#2}
3524     }
3525   }
3526   %\sref@label@id@arg{\omdoc@sect@name~\@nameuse{the#1}}\omgroup@id
3527 }

```

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

`omgroup`

```

3528 \keys_define:nn { document-structure / omgroupp }{
3529   id          .str_set_x:N = \l_@@_omgroup_id_str,
3530   date        .str_set_x:N = \l_@@_omgroup_date_str,
3531   creators     .clist_set:N = \l_@@_omgroup_creators_clist,
3532   contributors .clist_set:N = \l_@@_omgroup_contributors_clist,
3533   srccite      .tl_set:N    = \l_@@_omgroup_srccite_tl,
3534   type         .tl_set:N    = \l_@@_omgroup_type_tl,
3535   short        .tl_set:N    = \l_@@_omgroup_short_tl,
3536   display      .tl_set:N    = \l_@@_omgroup_display_tl,
3537   intro        .tl_set:N    = \l_@@_omgroup_intro_tl,
3538   loadmodules  .bool_set:N  = \l_@@_omgroup_loadmodules_bool
3539 }
3540 \cs_new_protected:Nn \l_@@_omgroup_args:n {
3541   \str_clear:N \l_@@_omgroup_id_str
3542   \str_clear:N \l_@@_omgroup_date_str
3543   \clist_clear:N \l_@@_omgroup_creators_clist
3544   \clist_clear:N \l_@@_omgroup_contributors_clist
3545   \tl_clear:N \l_@@_omgroup_srccite_tl
3546   \tl_clear:N \l_@@_omgroup_type_tl
3547   \tl_clear:N \l_@@_omgroup_short_tl
3548   \tl_clear:N \l_@@_omgroup_display_tl
3549   \tl_clear:N \l_@@_omgroup_intro_tl
3550   \bool_set_false:N \l_@@_omgroup_loadmodules_bool
3551   \keys_set:nn { document-structure / omgroupp } { #1 }
3552 }

```

we define a switch for numbering lines and a hook for the beginning of groups: The `\at@begin@omgroup` macro allows customization. It is run at the beginning of the `omgroup`, i.e. after the section heading.

```

3553 \newif\if@mainmatter\@mainmattertrue
3554 \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.

```

3555 \keys_define:nn { document-structure / sectioning }{
3556   name      .str_set_x:N = \l_@@_sect_name_str   ,
3557   ref       .str_set_x:N = \l_@@_sect_ref_str     ,
3558   clear     .bool_set:N = \l_@@_sect_clear_bool  ,
3559   num       .bool_set:N = \l_@@_sect_num_bool    ,
3560 }
3561 \cs_new_protected:Nn \_@@_sect_args:n {
3562   \str_clear:N \l_@@_sect_name_str
3563   \str_clear:N \l_@@_sect_ref_str
3564   \bool_set_false:N \l_@@_sect_clear_bool
3565   \bool_set_false:N \l_@@_sect_num_bool
3566   \keys_set:nn { document-structure / sectioning } { #1 }
3567 }
3568 \newcommand\omdoc@sectioning[3][]{
3569   \_@@_sect_args:n {#1}
3570   \bool_if:NT \l_@@_sect_clear_bool { \cleardoublepage }
3571   \if@mainmatter% numbering not overridden by frontmatter, etc.
3572     \bool_if:NTF \l_@@_sect_num_bool {
3573       \omgroup@num{#2}{#3}
3574     }{
3575       \omgroup@nonum{#2}{#3}
3576     }
3577   \def\current@section@level{\omdoc@sect@name}
3578   \else
3579     \omgroup@nonum{#2}{#3}
3580   \fi
3581 }% 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.

```

3582 \newcommand\omgroup@redefine@addtocontents[1]{%
3583   %\edef\@import{#1}%
3584   %\@for\@I:=\@import\do{%
3585     %\edef\@path{\csname module@\@I @path\endcsname}%
3586     %\@ifundefined{tf@toc}\relax%
3587     %   {\protected@write\tf@toc}{\string\@requiremodules{\@path}}}%
3588     %\ifx\hyper@anchor\@undefined% hyperref.sty loaded?
3589     %\def\addcontentsline##1##2##3{%
3590       %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3591       %\else% hyperref.sty not loaded
3592       %\def\addcontentsline##1##2##3{%
3593         %\addtocontents{##1}{\protect\contentsline{##2}{\string\withusedmodules{##1}{##3}}{\thepage}}%
3594         %\fi
3595       }% 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.

```

3596 \int_new:N \l_document_structure_omgroup_level_int
3597 \newenvironment{omgroup}[2][]{% keys, title
3598 {
3599   \_@@_omgroup_args:n { #1 }%\sref@target%
3600   \int_incr:N \l_document_structure_omgroup_level_int

```

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.

```

3601 \bool_if:NT \l_@@_omgroup_loadmodules_bool {
3602   \omgroup@redefine@addtocontents{
3603     %\@ifundefined{module@id}\used@modules%
3604     %{\@ifundefined{module@module@id @path}{\used@modules}\module@id}
3605   }
3606 }

```

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

```

3607 \int_incr:N\l_document_structure_section_level_int
3608 \ifcase\l_document_structure_section_level_int
3609   \or\omdoc@sectioning[name=\omdoc@part@kw,clear,num]{part}{#2}
3610   \or\omdoc@sectioning[name=\omdoc@chapter@kw,clear,num]{chapter}{#2}
3611   \or\omdoc@sectioning[name=\omdoc@section@kw,num]{section}{#2}
3612   \or\omdoc@sectioning[name=\omdoc@subsection@kw,num]{subsection}{#2}
3613   \or\omdoc@sectioning[name=\omdoc@subsubsection@kw,num]{subsubsection}{#2}
3614   \or\omdoc@sectioning[name=\omdoc@paragraph@kw,ref=this \omdoc@paragraph@kw]{paragraph}{#2}
3615   \or\omdoc@sectioning[name=\omdoc@subparagraph@kw,ref=this \omdoc@subparagraph@kw]{subparagraph}{#2}
3616 \fi
3617 \at@begin{omgroup}[#1]\l_document_structure_section_level_int{#2}
3618 \stex_ref_new_doc_target:n\l_@@_omgroup_id_str
3619 }% for customization
3620 {
3621   \int_decr:N\l_document_structure_section_level_int
3622   \int_decr:N\l_document_structure_omgroup_level_int
3623 }

```

and finally, we localize the sections

```

3624 \newcommand\omdoc@part@kw{Part}
3625 \newcommand\omdoc@chapter@kw{Chapter}
3626 \newcommand\omdoc@section@kw{Section}
3627 \newcommand\omdoc@subsection@kw{Subsection}
3628 \newcommand\omdoc@subsubsection@kw{Subsubsection}
3629 \newcommand\omdoc@paragraph@kw{paragraph}
3630 \newcommand\omdoc@subparagraph@kw{subparagraph}

```

28.7 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`

```

3631 \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).

```

3632 \cs_if_exist:NTF\frontmatter{
3633   \let\__orig_frontmatter\frontmatter

```

```

3634 \let\frontmatter\relax
3635 }{
3636 \tl_set:Nn\__orig_frontmatter{
3637 \clearpage
3638 \@mainmatterfalse
3639 \pagenumbering{roman}
3640 }
3641 }
3642 \cs_if_exist:NTF\backmatter{
3643 \let\__orig_backmatter\backmatter
3644 \let\backmatter\relax
3645 }{
3646 \tl_set:Nn\__orig_backmatter{
3647 \clearpage
3648 \@mainmatterfalse
3649 \pagenumbering{roman}
3650 }
3651 }

```

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.

```

3652 \newenvironment{frontmatter}{
3653 \__orig_frontmatter
3654 }{
3655 \cs_if_exist:NTF\mainmatter{
3656 \mainmatter
3657 }{
3658 \clearpage
3659 \@mainmattertrue
3660 \pagenumbering{arabic}
3661 }
3662 }

```

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

```

3663 \newenvironment{backmatter}{
3664 \__orig_backmatter
3665 }{
3666 \cs_if_exist:NTF\mainmatter{
3667 \mainmatter
3668 }{
3669 \clearpage
3670 \@mainmattertrue
3671 \pagenumbering{arabic}
3672 }
3673 }

```

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

```

3674 \@mainmattertrue\pagenumbering{arabic}

```

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


```

3675 \newcommand\afterprematurestop{}
3676 \def\prematurestop@endomgroup{
3677   \int_compare:nNnF \l_document_structure_omgroup_level_int = 0 {
3678     \end{omgroup}
3679     \int_decr:N \l_document_structure_omgroup_level_int
3680     \prematurestop@endomgroup
3681   }
3682 }
3683 \providecommand\prematurestop{
3684   \message{Stopping sTeX processing prematurely}
3685   \prematurestop@endomgroup
3686   \afterprematurestop
3687   \end{document}
3688 }

```

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

28.8 Global Variables

\setSGvar set a global variable

```

3689 \RequirePackage{etoolbox}
3690 \newcommand\setSGvar[1]{\@namedef{sTeX@Gvar@#1}}

```

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

\useSGvar use a global variable

```

3691 \newrobustcmd\useSGvar[1]{%
3692   \@ifundefined{sTeX@Gvar@#1}
3693   {\PackageError{omdoc}
3694     {The sTeX Global variable #1 is undefined}
3695     {set it with \protect\setSGvar}}
3696   \@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.

```

3697 \newrobustcmd\ifSGvar[3]{\def\@test{#2}%
3698   \@ifundefined{sTeX@Gvar@#1}
3699   {\PackageError{omdoc}
3700     {The sTeX Global variable #1 is undefined}
3701     {set it with \protect\setSGvar}}
3702   {\expandafter\ifx\csname sTeX@Gvar@#1\endcsname\@test #3\fi}}

```

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

Chapter 29

MiKoSlides – Implementation

29.1 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.

```
3703 \*cls)
3704 \@@=mikoslides)
3705 \ProvidesExplClass{mikoslides}{2020/12/06}{1.3}{MiKo slides Class}
3706 \RequirePackage{l3keys2e,expl-keystr-compatible}
3707
3708 \keys_define:nn{mikoslides / cls}{
3709   class .code:n = {
3710     \PassOptionsToClass{\CurrentOption}{omdoc}
3711     \str_if_eq:nnT{#1}{book}{
3712       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
3713     }
3714     \str_if_eq:nnT{#1}{report}{
3715       \PassOptionsToPackage{defaulttopsec=part}{mikoslides}
3716     }
3717   },
3718   notes .bool_set:N = \c__mikoslides_notes_bool ,
3719   slides .code:n = { \bool_set_false:N \c__mikoslides_notes_bool },
3720   unknown .code:n = {
3721     \PassOptionsToClass{\CurrentOption}{omdoc}
3722     \PassOptionsToClass{\CurrentOption}{beamer}
3723     \PassOptionsToPackage{\CurrentOption}{mikoslides}
3724   }
3725 }
3726 \ProcessKeysOptions{ mikoslides / cls }
3727 \bool_if:NTF \c__mikoslides_notes_bool {
3728   \PassOptionsToPackage{notes=true}{mikoslides}
3729 }{
3730   \PassOptionsToPackage{notes=false}{mikoslides}
3731 }
3732 \</cls)
```

now we do the same for the mikoslides package.

```

3733 <*package>
3734 \ProvidesExplPackage{mikoslides}{2020/12/06}{1.3}{MiKo slides Package}
3735 \RequirePackage{l3keys2e,expl-keystr-compat}
3736
3737 \keys_define:nn{mikoslides / pkg}{
3738   topsect          .str_set_x:N = \c__mikoslides_topsect_str,
3739   defaulttopsect   .str_set_x:N = \c__mikoslides_defaulttopsec_str,
3740   notes            .bool_set:N = \c__mikoslides_notes_bool ,
3741   slides           .code:n       = { \bool_set_false:N \c__mikoslides_notes_bool },
3742   sectocframes     .bool_set:N = \c__mikoslides_sectocframes_bool ,
3743   frameimages      .bool_set:N = \c__mikoslides_frameimages_bool ,
3744   fiboxed          .bool_set:N = \c__mikoslides_fiboxed_bool ,
3745   noproblems       .bool_set:N = \c__mikoslides_noproblems_bool,
3746   unknown          .code:n       = {
3747     \PassOptionsToClass{\CurrentOption}{stex}
3748     \PassOptionsToClass{\CurrentOption}{tikzinput}
3749   }
3750 }
3751 \ProcessKeysOptions{ mikoslides / pkg }
3752
```

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

```

3753 \str_if_empty:NTF \c__mikoslides_topsect_str {
3754   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_defaulttopsec_str
3755 }{
3756   \str_set_eq:NN \__mikoslidestopsect \c__mikoslides_topsect_str
3757 }
3758 </package>

```

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

```

3759 <*cls>
3760 \bool_if:NTF \c__mikoslides_notes_bool {
3761   \LoadClass{omdoc}
3762 }{
3763   \LoadClass[10pt,notheorems,xcolor={dvipsnames,svgnames}]{beamer}
3764   \newcounter{Item}
3765   \newcounter{paragraph}
3766   \newcounter{subparagraph}
3767   \newcounter{Hfootnote}
3768   \RequirePackage{omdoc}
3769 }

```

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

```

3770 \RequirePackage{mikoslides}
3771 </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 `TeX` packages. The first batch of packages we want are loaded on `mikoslides.sty`. These are the general ones, we will load the `TeX`-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.

```

3772 \*package>
3773 \RequirePackage{stex-compatibility}
3774 \RequirePackage{stex-tikzinput}
3775 \bool_if:NT \c__mikosslides_notes_bool {
3776   \RequirePackage{a4wide}
3777   \RequirePackage{marginnote}
3778   \PassOptionsToPackage{dvipsnames,svgnames}{xcolor}
3779   \RequirePackage{mdframed}
3780   \RequirePackage[noxcolor,noamsthm]{beamerarticle}
3781   \RequirePackage[bookmarks,bookmarksopen,bookmarksnumbered,breaklinks,hidelinks]{hyperref}
3782 }
3783 \RequirePackage{etoolbox}
3784 \RequirePackage{amssymb}
3785 \RequirePackage{amsmath}
3786 \RequirePackage{comment}
3787 \RequirePackage{textcomp}
3788 \RequirePackage{url}
3789 \RequirePackage{graphicx}
3790 \RequirePackage{pgf}

```

29.2 Notes and Slides

For the lecture notes cases, we also provide the `\usetheme` macro that would otherwise come from the the `beamer` class. While the latter loads `beamertheme<theme>.sty`, the notes version loads `beamernotestheme<theme>.sty`.¹⁰

```

3791 \bool_if:NT \c__mikosslides_notes_bool {
3792   \renewcommand\usetheme[2] [] {\usepackage[#1]{beamernotestheme#2}}
3793 }

```

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

```

3794 \newcounter{slide}
3795 \newlength{\slidewidth}\setlength{\slidewidth}{13.5cm}
3796 \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.

```

3797 \bool_if:NTF \c__mikosslides_notes_bool {
3798   \renewenvironment{note}{\ignorespaces}{}}
3799 }{
3800   \excludecomment{note}
3801 }

```

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.

```

3802 \bool_if:NT \c__mikosslides_notes_bool {
3803   \newlength{\slideframewidth}
3804   \setlength{\slideframewidth}{1.5pt}

```

¹⁰EDNOTE: MK: This is not ideal, but I am not sure that I want to be able to provide the full theme functionality there.

frame We first define the keys.

```

3805 \cs_new_protected:Nn \__mikoslides_do_yes_param:Nn {
3806   \exp_args:Nx \str_if_eq:nnTF { \str_uppercase:n{ #2 } }{ yes }{
3807     \bool_set_true:N #1
3808   }{
3809     \bool_set_false:N #1
3810   }
3811 }
3812 \keys_define:nn{mikoslides / frame}{
3813   label .str_set_x:N = \l__mikoslides_frame_label_str,
3814   allowframebreaks .code:n = {
3815     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowframebreaks_bool { #1 }
3816   },
3817   allowdisplaybreaks .code:n = {
3818     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_allowdisplaybreaks_bool { #1 }
3819   },
3820   fragile .code:n = {
3821     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_fragile_bool { #1 }
3822   },
3823   shrink .code:n = {
3824     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_shrink_bool { #1 }
3825   },
3826   squeeze .code:n = {
3827     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_squeeze_bool { #1 }
3828   },
3829   t .code:n = {
3830     \__mikoslides_do_yes_param:Nn \l__mikoslides_frame_t_bool { #1 }
3831   },
3832 }
3833 \cs_new_protected:Nn \__mikoslides_frame_args:n {
3834   \str_clear:N \l__mikoslides_frame_label_str
3835   \bool_set_true:N \l__mikoslides_frame_allowframebreaks_bool
3836   \bool_set_true:N \l__mikoslides_frame_allowdisplaybreaks_bool
3837   \bool_set_true:N \l__mikoslides_frame_fragile_bool
3838   \bool_set_true:N \l__mikoslides_frame_shrink_bool
3839   \bool_set_true:N \l__mikoslides_frame_squeeze_bool
3840   \bool_set_true:N \l__mikoslides_frame_t_bool
3841   \keys_set:nn { mikoslides / frame }{ #1 }
3842 }

```

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

```

3843 \renewenvironment{frame}[1][]{
3844   \__mikoslides_frame_args:n{#1}
3845   \sffamily
3846   \stepcounter{slide}
3847   \def\@currentlabel{\theslide}
3848   \str_if_empty:NF \l__mikoslides_frame_label_str {
3849     \label{\l__mikoslides_frame_label_str}
3850   }

```

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

```

3851 \def\itemize@level{outer}
3852 \def\itemize@outer{outer}
3853 \def\itemize@inner{inner}
3854 \renewcommand\newpage{\addtocounter{framenumber}{1}}

```

```

3855 \renewcommand\metakeys@show@keys[2]{\marginnote{\scriptsize ##2}}
3856 \renewenvironment{itemize}{
3857   \ifx\itemize@level\itemize@outer
3858     \def\itemize@label{\$ \rhd\$}
3859   \fi
3860   \ifx\itemize@level\itemize@inner
3861     \def\itemize@label{\$ \scriptstyle \rhd\$}
3862   \fi
3863   \begin{list}
3864     {\itemize@label}
3865     {\setlength{\labelsep}{.3em}
3866      \setlength{\labelwidth}{.5em}
3867      \setlength{\leftmargin}{1.5em}
3868     }
3869   \edef\itemize@level{\itemize@inner}
3870 }{
3871   \end{list}
3872 }

```

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

```

3873 \begin{mdframed}[linewidth=\slideframewidth,skipabove=1ex,skipbelow=1ex,userdefinedwidth]
3874 }{
3875   \medskip\miko@slidelabel\end{mdframed}
3876 }

```

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

\frametitle

```

3877 \renewcommand{\frametitle}[1]{\Large\bf\sf\color{blue}{#1}}\medskip
3878 }

```

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

EdN:11

\pause 11

```

3879 \bool_if:NT \c__mikoslides_notes_bool {
3880   \newcommand\pause{}
3881 }

```

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

nomtext

```

3882 \bool_if:NTF \c__mikoslides_notes_bool {
3883   \newenvironment{nomtext}[1][\begin{omtext}[#1]}\end{omtext}}
3884 }{
3885   \excludecomment{nomtext}
3886 }

```

nomgroup

```

3887 \bool_if:NTF \c__mikoslides_notes_bool {
3888   \newenvironment{nomgroup}[2][\begin{omgroup}[#1][#2]}\end{omgroup}}
3889 }{
3890   \excludecomment{nomgroup}
3891 }

```

¹¹EdNOTE: MK: fake it in notes mode for now

ndefinition

```
3892 \bool_if:NTF \c__mikoslides_notes_bool {
3893   \newenvironment{ndefinition}[1] [] {\begin{definition}[#1]}{\end{definition}}
3894 }{
3895   \excludecomment{ndefinition}
3896 }
```

nassertion

```
3897 \bool_if:NTF \c__mikoslides_notes_bool {
3898   \newenvironment{nassertion}[1] [] {\begin{assertion}[#1]}{\end{assertion}}
3899 }{
3900   \excludecomment{nassertion}
3901 }
```

nsproof

```
3902 \bool_if:NTF \c__mikoslides_notes_bool {
3903   \newenvironment{nsproof}[2] [] {\begin{sproof}[#1]{#2}}{\end{sproof}}
3904 }{
3905   \excludecomment{nsproof}
3906 }
```

nexample

```
3907 \bool_if:NTF \c__mikoslides_notes_bool {
3908   \newenvironment{nexample}[1] [] {\begin{example}[#1]}{\end{example}}
3909 }{
3910   \excludecomment{nexample}
3911 }
```

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

```
3912 \def\inputref@preskip{\smallskip}
3913 \def\inputref@postskip{\medskip}
```

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

\inputref*

```
3914 \let\orig@inputref\inputref
3915 \def\inputref{\@ifstar\ninputref\orig@inputref}
3916 \newcommand\ninputref[2] [] {
3917   \bool_if:NT \c__mikoslides_notes_bool {
3918     \orig@inputref[#1]{#2}
3919   }
3920 }
```

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

29.3 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.

`\setslidelo` The default logo is the \TeX logo. Customization can be done by `\setslidelo{<logo name>}`.

```

3921 \newlength{\slideloheight}
3922
3923 \bool_if:NTF \c_mikoslides_notes_bool {
3924   \setlength{\slideloheight}{.4cm}
3925 }{
3926   \setlength{\slideloheight}{1cm}
3927 }
3928 \newsavebox{\slidelo}
3929 \sbox{\slidelo}{\TeX}
3930 \newrobustcmd{\setslidelo}[1]{
3931   \sbox{\slidelo}{\includegraphics[height=\slideloheight]{#1}}
3932 }

```

(End definition for `\setslidelo`. 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.

```

3933 \def\source{Michael Kohlhase}% customize locally
3934 \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.

```

3935 \def\copyrightnotice{\footnotesize\copyright:\hspace{.3ex}{\source}}
3936 \newsavebox{\cclogo}
3937 \sbox{\cclogo}{\includegraphics[height=\slideloheight]{cc_somerights}}
3938 \newif\ifcchref\cchreffalse
3939 \AtBeginDocument{
3940   \ifpackageloaded{hyperref}{\cchreftrue}{\cchreffalse}
3941 }
3942 \def\licensing{
3943   \ifcchref
3944     \href{http://creativecommons.org/licenses/by-sa/2.5/}{\usebox{\cclogo}}
3945   \else
3946     {\usebox{\cclogo}}
3947   \fi
3948 }
3949 \newrobustcmd{\setlicensing}[2][]{
3950   \def\@url{#1}
3951   \sbox{\cclogo}{\includegraphics[height=\slideloheight]{#2}}
3952   \ifx\@url\@empty
3953     \def\licensing{{\usebox{\cclogo}}}
3954   \else
3955     \def\licensing{
3956       \ifcchref
3957         \href{#1}{\usebox{\cclogo}}
3958       \else
3959         {\usebox{\cclogo}}
3960       \fi

```



```

3961     }
3962     \fi
3963 }

```

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

EdN:12

`\slidelabel` Now, we set up the slide label for the article mode.¹²

```

3964 \newrobustcmd\miko@slidelabel{
3965   \vbox to \slidelogoheight{
3966     \vss\hbox to \slidewidth
3967     {\licensing\hfill\copyrightnotice\hfill\arabic{slide}\hfill\usebox{\slidelogo}}
3968   }
3969 }

```

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

29.4 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.

```

3970 \def\Gin@mhrepos{}
3971 \define@key{Gin}{mhrepos}{\def\Gin@mhrepos{#1}}
3972 \define@key{Gin}{label}{\def\currentlabel{\arabic{slide}}\label{#1}}
3973 \newrobustcmd\frameimage[2][]{
3974   \stepcounter{slide}
3975   \bool_if:NT \c__mikoslides_frameimages_bool {
3976     \def\Gin@ewidth{}\setkeys{Gin}{#1}
3977     \bool_if:NF \c__mikoslides_notes_bool { \vfill }
3978     \begin{center}
3979       \bool_if:NTF \c__mikoslides_fboxed_bool {
3980         \fbox{
3981           \ifx\Gin@ewidth\@empty
3982             \ifx\Gin@mhrepos\@empty
3983               \mhgraphics[width=\slidewidth,#1]{#2}
3984             \else
3985               \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
3986             \fi
3987           \else% Gin@ewidth empty
3988             \ifx\Gin@mhrepos\@empty
3989               \mhgraphics[#1]{#2}
3990             \else
3991               \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
3992             \fi
3993           \fi% Gin@ewidth empty
3994         }
3995       }{
3996         \ifx\Gin@ewidth\@empty
3997           \ifx\Gin@mhrepos\@empty
3998             \mhgraphics[width=\slidewidth,#1]{#2}
3999           \else
4000             \mhgraphics[width=\slidewidth,#1,mhrepos=\Gin@mhrepos]{#2}
4001           \fi

```

¹²EdNOTE: see that we can use the themes for the slides some day. This is all fake.

```

4002         \ifx\Gin@mhrepos\@empty
4003             \mhgraphics[#1]{#2}
4004         \else
4005             \mhgraphics[#1,mhrepos=\Gin@mhrepos]{#2}
4006         \fi
4007     \fi% Gin@ewidth empty
4008 }
4009 \end{center}
4010 \par\strut\hfill{\footnotesize Slide \arabic{slide}}}%
4011 \bool_if:NF \c__mikoslides_notes_bool { \vfill }
4012 }
4013 } % ifmks@sty@frameimages

```

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

29.5 Colors and Highlighting

We first specify sans serif fonts as the default.

```

4014 \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

```

4015 \AddToHook{begindocument}{
4016     \definecolor{green}{rgb}{0,.5,0}
4017     \definecolor{purple}{cmyk}{.3,1,0,.17}
4018 }

```

We customize the `\defemph`, `\symrefemph`, `\compemph`, and `\titleemph` macros with colors. Furthermore we customize the `\@@lec` macro for the appearance of line end comments in `\lec`.

```

4019 % \def\STpresent#1{\textcolor{blue}{#1}}
4020 \def\defemph#1{\textcolor{magenta}{#1}}
4021 \def\symrefemph#1{\textcolor{cyan}{#1}}
4022 \def\compemph#1{\textcolor{blue}{#1}}
4023 \def\titleemph#1{\textcolor{blue}{#1}}
4024 \def\__mikoslideslec#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.

```

4025 \pgfdeclareimage[width=.8em]{miko@small@dbend}{dangerous-bend}
4026 \def\smalltextwarning{
4027     \pgfuseimage{miko@small@dbend}
4028     \xspace
4029 }
4030 \pgfdeclareimage[width=1.2em]{miko@dbend}{dangerous-bend}
4031 \newrobustcmd\textwarning{
4032     \raisebox{-.05cm}{\pgfuseimage{miko@dbend}}
4033     \xspace
4034 }
4035 \pgfdeclareimage[width=2.5em]{miko@big@dbend}{dangerous-bend}

```

```

4036 \newrobustcmd\bigtextwarning{
4037   \raisebox{-.05cm}{\pgfuseimage{miko@big@dbend}}
4038   \xspace
4039 }

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

4040 \newrobustcmd\putgraphicsat[3]{
4041   \begin{picture}(0,0)\put(#1){\includegraphics[#2]{#3}}\end{picture}
4042 }
4043 \newrobustcmd\putat[2]{
4044   \begin{picture}(0,0)\put(#1){#2}\end{picture}
4045 }

```

29.6 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`.

```

4046 \bool_if:NT \c__mikoslides_sectocframes_bool {
4047   \str_if_eq:VnTF \__mikoslidestopsect{part}{
4048     \newcounter{chapter}\counterwithin*{section}{chapter}
4049   }{
4050     \str_if_eq:VnT\__mikoslidestopsect{chapter}{
4051       \newcounter{chapter}\counterwithin*{section}{chapter}
4052     }
4053   }
4054 }

```

`\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

4055 \str_case:VnF \__mikoslidestopsect {
4056   {part}{
4057     \int_set:Nn \l_document_structure_section_level_int {0}
4058     \def\thesection{\arabic{chapter}.\arabic{section}}
4059     \def\part@prefix{\arabic{chapter}.}
4060   }
4061   {chapter}{
4062     \int_set:Nn \l_document_structure_section_level_int {1}
4063     \def\thesection{\arabic{chapter}.\arabic{section}}
4064     \def\part@prefix{\arabic{chapter}.}
4065   }
4066 }{
4067   \int_set:Nn \l_document_structure_section_level_int {2}
4068   \def\part@prefix{}
4069 }
4070
4071 \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

```

4072 \renewenvironment{omgroup}[2][]{
4073   \_document_structure_omgroup_args:n { #1 }
4074   \int_incr:N \l_document_structure_omgroup_level_int
4075   \int_incr:N \l_document_structure_section_level_int
4076   \bool_if:NT \c__mikosslides_sectocframes_bool {
4077     \stepcounter{slide}
4078     \begin{frame}[noframenumbering]
4079     \vfill\Large\centering
4080     \red{
4081       \ifcase\l_document_structure_section_level_int\or
4082         \stepcounter{part}
4083         \def\__mikosslideslabel{\omdoc@part@kw~\Roman{part}}
4084         \def\currentsectionlevel{\omdoc@part@kw}
4085       \or
4086         \stepcounter{chapter}
4087         \def\__mikosslideslabel{\omdoc@chapter@kw~\arabic{chapter}}
4088         \def\currentsectionlevel{\omdoc@chapter@kw}
4089       \or
4090         \stepcounter{section}
4091         \def\__mikosslideslabel{\part@prefix\arabic{section}}
4092         \def\currentsectionlevel{\omdoc@section@kw}
4093       \or
4094         \stepcounter{subsection}
4095         \def\__mikosslideslabel{\part@prefix\arabic{section}.\arabic{subsection}}
4096         \def\currentsectionlevel{\omdoc@subsection@kw}
4097       \or
4098         \stepcounter{subsubsection}
4099         \def\__mikosslideslabel{\part@prefix\arabic{section}.\arabic{subsection}.\arabic{subsubsection}}
4100         \def\currentsectionlevel{\omdoc@subsubsection@kw}
4101       \or
4102         \stepcounter{mparagraph}
4103         \def\__mikosslideslabel{\part@prefix\arabic{section}.\arabic{msubsection}.\arabic{mparagraph}}
4104         \def\currentsectionlevel{\omdoc@paragraph@kw}
4105       \fi% end ifcase
4106       \__mikosslideslabel\sref@label@id\__mikosslideslabel
4107       \quad #2%
4108     }%
4109     \vfill%
4110     \end{frame}%
4111   }
4112   \stex_ref_new_doc_target:n\l_document_structure_omgroup_id_str%
4113 }{
4114   \int_decr:N \l_document_structure_section_level_int
4115 }
4116 }

```

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

```

4117 \def\inserttheorembodyfont{\normalfont}
4118 \bool_if:NF \c__mikosslides_notes_bool {
4119   \defbeamertemplate{theorem begin}{miko}
4120   {\inserttheoremheadfont\inserttheoremname\inserttheoremnumber
4121     \ifx\inserttheoremaddition\@empty\else\ (\inserttheoremaddition)\fi%

```

```

4122 \inserttheorempunctuation\inserttheorembodyfont\xspace}
4123 \defbeamertemplate{theorem end}{miko}{}

```

and we set it as the default one.

```

4124 \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.

```

4125 \expandafter\def\csname Parent2\endcsname{}
4126 }
4127 \bool_if:Nt \c__mikoslides_notes_bool {
4128 \renewenvironment{columns}[1][]{%
4129 \par\noindent%
4130 \begin{minipage}%
4131 \slidewidth\centering\leavevmode%
4132 }{%
4133 \end{minipage}\par\noindent%
4134 }%
4135 \newsavebox\columnbox%
4136 \renewenvironment<>{column}[2][]{%
4137 \begin{lrbox}{\columnbox}\begin{minipage}{#2}%
4138 }{%
4139 \end{minipage}\end{lrbox}\usebox\columnbox%
4140 }%
4141 }
4142 \bool_if:NTF \c__mikoslides_noproblems_bool {
4143 \newenvironment{problems}{}{}
4144 }{
4145 \excludecomment{problems}
4146 }

```

29.7 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.

```

4147 \gdef\printexcursions{}
4148 \newcommand\excursionref[2]{% label, text
4149 \bool_if:NT \c__mikoslides_notes_bool {
4150 \begin{omtext}[title=Excursion]
4151 #2 \sref[fallback=the appendix]{#1}.
4152 \end{omtext}
4153 }
4154 }
4155 \newcommand\activate@excursion[2][]{
4156 \gappto\printexcursions{\inputref{#1}{#2}}
4157 }
4158 \newcommand\excursion[4][]{% repos, label, path, text
4159 \bool_if:NT \c__mikoslides_notes_bool {
4160 \activate@excursion{#1}{#3}\excursionref{#2}{#4}
4161 }
4162 }

```

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

\excursiongroup

```

4163 \keys_define:nn{mikoslides / excursiongroup }{
4164   id          .str_set_x:N = \l__mikoslides_excursion_id_str,
4165   intro       .tl_set:N    = \l__mikoslides_excursion_intro_tl,
4166   mhrepos     .str_set_x:N = \l__mikoslides_excursion_mhrepos_str
4167 }
4168 \cs_new_protected:Nn \__mikoslides_excursion_args:n {
4169   \tl_clear:N \l__mikoslides_excursion_intro_tl
4170   \str_clear:N \l__mikoslides_excursion_id_str
4171   \str_clear:N \l__mikoslides_excursion_mhrepos_str
4172   \keys_set:nn {mikoslides / excursiongroup }{ #1 }
4173 }
4174 \newcommand\excursiongroup[1][ ]{
4175   \__mikoslides_excursion_args:n{ #1 }
4176   \ifdefempty\printexcursions{}% only if there are excursions
4177   {
4178     \begin{omgroup}[#1]{Excursions}%
4179     \ifdefempty\l__mikoslides_excursion_intro_tl{{
4180       \inputref[\l__mikoslides_excursion_mhrepos_str]{
4181         \l__mikoslides_excursion_intro_tl
4182       }
4183     }
4184     \printexcursions%
4185     \end{omgroup}
4186   }
4187 }
4188 \</package>

```

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

Chapter 30

The Implementation

30.1 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.

```
4189 <*package>
4190 <@@=problems>
4191 \ProvidesExplPackage{problem}{2019/03/20}{1.3}{Semantic Markup for Problems}
4192 \RequirePackage{l3keys2e,expl-keystr-compat}
4193
4194 \keys_define:nn { problem / pkg }{
4195   notes      .default:n    = { true },
4196   notes      .bool_set:N   = \c__problems_notes_bool,
4197   gnotes     .default:n    = { true },
4198   gnotes     .bool_set:N   = \c__problems_gnotes_bool,
4199   hints      .default:n    = { true },
4200   hints      .bool_set:N   = \c__problems_hints_bool,
4201   solutions  .default:n    = { true },
4202   solutions  .bool_set:N   = \c__problems_solutions_bool,
4203   pts        .default:n    = { true },
4204   pts        .bool_set:N   = \c__problems_pts_bool,
4205   min        .default:n    = { true },
4206   min        .bool_set:N   = \c__problems_min_bool,
4207   boxed      .default:n    = { true },
4208   boxed      .bool_set:N   = \c__problems_boxed_bool
4209 }
4210 \def\solutionstrue{
4211   \bool_set_true:N \c__problems_solutions_bool
4212 }
4213 \def\solutionsfalse{
4214   \bool_set_false:N \c__problems_solutions_bool
4215 }
4216
4217 \ProcessKeysOptions{ problem / pkg }
4218
4219 \RequirePackage{stex-compatibility}
```

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

```
4219 \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.

```
4220 \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.

```
4221 \def\prob@problem@kw{Problem}
4222 \def\prob@solution@kw{Solution}
4223 \def\prob@hint@kw{Hint}
4224 \def\prob@note@kw{Note}
4225 \def\prob@gnote@kw{Grading}
4226 \def\prob@pt@kw{pt}
4227 \def\prob@min@kw{min}
```

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

For the other languages, we set up triggers

```
4228 \@ifpackageloaded{babel}{
4229   \clist_set:Nx \l_tmpa_clist {\bbl@loaded}
4230   \clist_if_in:NnT \l_tmpa_clist {ngerman}{
4231     \input{problem-ngerman.ldf}
4232   }
4233   \clist_if_in:NnT \l_tmpa_clist {finnish}{
4234     \input{problem-finnish.ldf}
4235   }
4236   \clist_if_in:NnT \l_tmpa_clist {french}{
4237     \input{problem-french.ldf}
4238   }
4239   \clist_if_in:NnT \l_tmpa_clist {russian}{
4240     \input{problem-russian.ldf}
4241   }
4242 }{}

```

30.2 Problems and Solutions

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

```
4243 \keys_define:nn{ problem / problem }{
4244   id      .str_set:x:N = \l__problems_prob_id_str,
4245   pts     .tl_set:N    = \l__problems_prob_pts_tl,
4246   min     .tl_set:N    = \l__problems_prob_min_tl,
4247   title   .tl_set:N    = \l__problems_prob_title_tl,
4248   refnum  .int_set:N    = \l__problems_prob_refnum_int
4249 }
4250 \cs_new_protected:Nn \__problems_prob_args:n {
4251   \str_clear:N \l__problems_prob_id_str
4252   \tl_clear:N \l__problems_prob_pts_tl
4253   \tl_clear:N \l__problems_prob_min_tl
4254   \tl_clear:N \l__problems_prob_title_tl
4255   \int_zero_new:N \l__problems_prob_refnum_int

```



```

4256 \keys_set:nn { problem / problem }{ #1 }
4257 }

```

Then we set up a counter for problems.

`\numberproblemsin`

```

4258 \newcounter{problem}
4259 \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.

```

4260 \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

```

4261 \newcommand\prob@number{
4262   \int_if_exist:NTF \l__problems_inclprob_refnum_int {
4263     \prob@label{\int_use:N \l__problems_inclprob_refnum_int }
4264   }{
4265     \int_if_exist:NTF \l__problems_prob_refnum_int {
4266       \prob@label{\int_use:N \l__problems_prob_refnum_int }
4267     }{
4268       \prob@label\theproblem
4269     }
4270   }
4271 }

```

(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.

```

4272 \newcommand\prob@title[3]{%
4273   \tl_if_exist:NTF \l__problems_inclprob_title_tl {
4274     #2 \l__problems_inclprob_title_tl #3
4275   }{
4276     \tl_if_exist:NTF \l__problems_prob_title_tl {
4277       #2 \l__problems_prob_title_tl #3
4278     }{
4279       #1
4280     }
4281   }
4282 }

```

(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.

```

4283 \def\prob@heading{
4284   \prob@problem@kw~\prob@number\prob@title{ }{ (}{) \strut}
4285   %\sref@label{id{\prob@problem@kw~\prob@number}}{ }
4286 }

```

(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`

```

4287 \newenvironment{problem}[1][]{
4288   \__problems_prob_args:n{#1}%\sref@target%
4289   \@in@omtexttrue% we are in a statement (for inline definitions)
4290   \stepcounter{problem}\record@problem
4291   \def\current@section@level{\prob@problem@kw}
4292   \par\noindent\textbf{\prob@heading\show@pts\show@min\\ignorespacesandpars
4293 }%
4294   {\smallskip}
4295   \bool_if:NT \c__problems_boxed_bool {
4296     \surroundwithmdframed{problem}
4297   }

```

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

```

4298 \def\record@problem{
4299   \protected@write\@auxout{}
4300   {
4301     \string\@problem{\prob@number}
4302     {
4303       \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4304         \l__problems_inclprob_pts_tl
4305       }{
4306         \l__problems_prob_pts_tl
4307       }
4308     }%
4309     {
4310       \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4311         \l__problems_inclprob_min_tl
4312       }{
4313         \l__problems_prob_min_tl
4314       }
4315     }
4316   }
4317 }

```

(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).

```

4318 \def\@problem#1#2#3{}

```

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

`solution`

The `solution` environment is similar to the `problem` environment, only that it is independent of the boxed mode. It also has it's own keys that we need to define first.

```

4319 \keys_define:nn { problem / solution }{
4320   id          .str_set_x:N = \l__problems_solution_id_str ,
4321   for         .tl_set:N    = \l__problems_solution_for_tl ,

```

```

4322 height      .dim_set:N    = \l__problems_solution_height_dim ,
4323 creators     .clist_set:N  = \l__problems_solution_creators_clist ,
4324 contributors  .clist_set:N  = \l__problems_solution_contributors_clist ,
4325 srccite      .tl_set:N     = \l__problems_solution_srccite_tl
4326 }
4327 \cs_new_protected:Nn \__problems_solution_args:n {
4328   \str_clear:N \l__problems_solution_id_str
4329   \tl_clear:N \l__problems_solution_for_tl
4330   \tl_clear:N \l__problems_solution_srccite_tl
4331   \clist_clear:N \l__problems_solution_creators_clist
4332   \clist_clear:N \l__problems_solution_contributors_clist
4333   \dim_zero:N \l__problems_solution_height_dim
4334   \keys_set:nn { problem / solution }{ #1 }
4335 }

```

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

```

4336 \newcommand\@startsolution[1][]{
4337   \__problems_solution_args:n { #1 }
4338   \@in@omtexttrue% we are in a statement.
4339   \bool_if:NF \c__problems_boxed_bool { \hrule }
4340   \smallskip\noindent
4341   {\textbf\prob@solution@kw : \enspace}
4342   \begin{small}
4343   \def\current@section@level{\prob@solution@kw}
4344   \ignorespacesandpars
4345 }

```

\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.

```

4346 \newcommand\startsolutions{
4347   \specialcomment{solution}{\@startsolution}{
4348     \bool_if:NF \c__problems_boxed_bool {
4349       \hrule\medskip
4350     }
4351     \end{small}%
4352   }
4353   \bool_if:NT \c__problems_boxed_bool {
4354     \surroundwithmdframed{solution}
4355   }
4356 }

```

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

\stopsolutions

```

4357 \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.

```

4358 \bool_if:NTF \c__problems_solutions_bool {
4359   \startsolutions
4360 }{
4361   \stopsolutions
4362 }

```

exnote

```
4363 \bool_if:NTF \c_problems_notes_bool {  
4364   \newenvironment{exnote}[1][]{  
4365     \par\smallskip\hrule\smallskip  
4366     \noindent\textbf{\prob@note@kw : }\small  
4367   }{  
4368     \smallskip\hrule  
4369   }  
4370 }{  
4371   \excludecomment{exnote}  
4372 }
```

hint

```
4373 \bool_if:NTF \c_problems_notes_bool {  
4374   \newenvironment{hint}[1][]{  
4375     \par\smallskip\hrule\smallskip  
4376     \noindent\textbf{\prob@hint@kw : }\small  
4377   }{  
4378     \smallskip\hrule  
4379   }  
4380   \newenvironment{exhint}[1][]{  
4381     \par\smallskip\hrule\smallskip  
4382     \noindent\textbf{\prob@hint@kw : }\small  
4383   }{  
4384     \smallskip\hrule  
4385   }  
4386 }{  
4387   \excludecomment{hint}  
4388   \excludecomment{exhint}  
4389 }
```

gnote

```
4390 \bool_if:NTF \c_problems_notes_bool {  
4391   \newenvironment{gnote}[1][]{  
4392     \par\smallskip\hrule\smallskip  
4393     \noindent\textbf{\prob@gnote@kw : }\small  
4394   }{  
4395     \smallskip\hrule  
4396   }  
4397 }{  
4398   \excludecomment{gnote}  
4399 }
```

30.3 Multiple Choice Blocks

EdN:13

mcb 13

```
4400 \newenvironment{mcb}{  
4401   \begin{enumerate}  
4402 }{  
4403   \end{enumerate}  
4404 }
```

¹³EdNOTE: MK: maybe import something better here from a dedicated MC package

we define the keys for the mcc macro

```

4405 \cs_new_protected:Nn \__problems_do_yes_param:Nn {
4406   \exp_args:Nx \str_if_eq:nnTF { \str_lowercase:n{ #2 } }{ yes }{
4407     \bool_set_true:N #1
4408   }{
4409     \bool_set_false:N #1
4410   }
4411 }
4412 \keys_define:nn { problem / mcc }{
4413   id      .str_set_x:N = \l__problems_mcc_id_str ,
4414   feedback .tl_set:N   = \l__problems_mcc_feedback_tl ,
4415   T       .default:n   = { true } ,
4416   T       .bool_set:N  = \l__problems_mcc_t_bool ,
4417   F       .default:n   = { true } ,
4418   F       .bool_set:N  = \l__problems_mcc_f_bool ,
4419   Ttext    .code:n     = {
4420     \__problems_do_yes_param:Nn \l__problems_mcc_Ttext_bool { #1 }
4421   } ,
4422   Ftext    .code:n     = {
4423     \__problems_do_yes_param:Nn \l__problems_mcc_Ftext_bool { #1 }
4424   }
4425 }
4426 \cs_new_protected:Nn \l__problems_mcc_args:n {
4427   \str_clear:N \l__problems_mcc_id_str
4428   \tl_clear:N \l__problems_mcc_feedback_tl
4429   \bool_set_true:N \l__problems_mcc_t_bool
4430   \bool_set_true:N \l__problems_mcc_f_bool
4431   \bool_set_true:N \l__problems_mcc_Ttext_bool
4432   \bool_set_false:N \l__problems_mcc_Ftext_bool
4433   \keys_set:nn { problem / mcc }{ #1 }
4434 }

```

\mcc

```

4435 \newcommand\mcc[2][]{
4436   \l__problems_mcc_args:n{ #1 }
4437   \item #2
4438   \bool_if:NT \c__problems_solutions_bool {
4439     \
4440     \bool_if:NT \l__problems_mcc_t_bool {
4441       % TODO!
4442       % \ifcsstring{mcc@T}{T}{\mcc@Ttext}%
4443     }
4444     \bool_if:NT \l__problems_mcc_f_bool {
4445       % TODO!
4446       % \ifcsstring{mcc@F}{F}{\mcc@Ftext}%
4447     }
4448     \tl_if_empty:NTF \l__problems_mcc_feedback_tl {
4449       !
4450     }{
4451       \l__problems_mcc_feedback_tl
4452     }
4453   }
4454 } %solutions

```

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

30.4 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.

```

4455
4456 \keys_define:nn{ problem / inclproblem }{
4457   % id      .str_set:x:N = \l__problems_inclprob_id_str,
4458   pts      .tl_set:N    = \l__problems_inclprob_pts_tl,
4459   min      .tl_set:N    = \l__problems_inclprob_min_tl,
4460   title    .tl_set:N    = \l__problems_inclprob_title_tl,
4461   refnum   .int_set:N    = \l__problems_inclprob_refnum_int
4462 }
4463 \cs_new_protected:Nn \__problems_inclprob_args:n {
4464   % \str_clear:N \l__problems_prob_id_str
4465   \tl_clear:N \l__problems_inclprob_pts_tl
4466   \tl_clear:N \l__problems_inclprob_min_tl
4467   \tl_clear:N \l__problems_inclprob_title_tl
4468   \int_zero_new:N \l__problems_inclprob_refnum_int
4469   \keys_set:nn { problem / inclproblem }{ #1 }
4470 }
4471
4472 \cs_new_protected:Nn \__problems_inclprob_clear: {
4473   % \str_clear:N \l__problems_prob_id_str
4474   \let\l__problems_inclprob_pts_tl\undefined
4475   \let\l__problems_inclprob_min_tl\undefined
4476   \let\l__problems_inclprob_title_tl\undefined
4477   \let\l__problems_inclprob_refnum_int\undefined
4478 }
4479
4480 \newcommand\includeproblem[2][ ]{
4481   \__problems_inclprob_args:n{ #1 }
4482   \edef\temp@path{#2}
4483   \if@iswindows@\path@to@windows\temp@path\fi %TODO ?
4484   \input{\temp@path}
4485   \__problems_inclprob:clear:
4486 }

```

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

30.5 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.

```

4487 \AddToHook{enddocument}{
4488   \bool_if:NT \c__problems_pts_bool {
4489     \message{Total:~\arabic{pts}~points}
4490   }
4491   \bool_if:NT \c__problems_min_bool {
4492     \message{Total:~\arabic{min}~minutes}
4493   }
4494 }

```

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

```

4495 \def\pts#1{
4496   \bool_if:NT \c__problems_pts_bool {
4497     \marginpar{#1~\prob@pt@kw}
4498   }
4499 }
4500 \def\min#1{
4501   \bool_if:NT \c__problems_min_bool {
4502     \marginpar{#1~\prob@min@kw}
4503   }
4504 }

```

`\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.

```

4505 \newcounter{pts}
4506 \def\show@pts{
4507   \tl_if_exist:NTF \l__problems_inclprob_pts_tl {
4508     \bool_if:NT \c__problems_pts_bool {
4509       \marginpar{\l__problems_inclprob_pts_tl;\prob@pt@kw\smallskip}
4510       \addtocounter{pts}{\l__problems_inclprob_pts_tl}
4511     }
4512   }{
4513     \tl_if_exist:NT \l__problems_prob_pts_tl {
4514       \bool_if:NT \c__problems_pts_bool {
4515         \marginpar{\l__problems_prob_pts_tl;\prob@pt@kw\smallskip}
4516         \addtocounter{pts}{\l__problems_prob_pts_tl}
4517       }
4518     }
4519   }
4520 }

```

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

`\show@min`

```

4521 \newcounter{min}
4522 \def\show@min{
4523   \tl_if_exist:NTF \l__problems_inclprob_min_tl {
4524     \bool_if:NT \c__problems_min_bool {
4525       \marginpar{\l__problems_inclprob_pts_tl;min}
4526       \addtocounter{min}{\l__problems_inclprob_min_tl}
4527     }
4528   }{
4529     \tl_if_exist:NT \l__problems_prob_min_tl {
4530       \bool_if:NT \c__problems_min_bool {
4531         \marginpar{\l__problems_prob_min_tl;min}
4532         \addtocounter{min}{\l__problems_prob_min_tl}
4533       }
4534     }
4535   }
4536 }
4537 \</package>

```

(End definition for `\show@min`. This function is documented on page ??.)